#122 DECEMBER 1986

2.95 (3.95 CANADA)

Dr. Dobb's Journal of Software Tools

FOR THE PROFESSIONAL PROGRAMMER

MULTITASKING

Programming Ethics

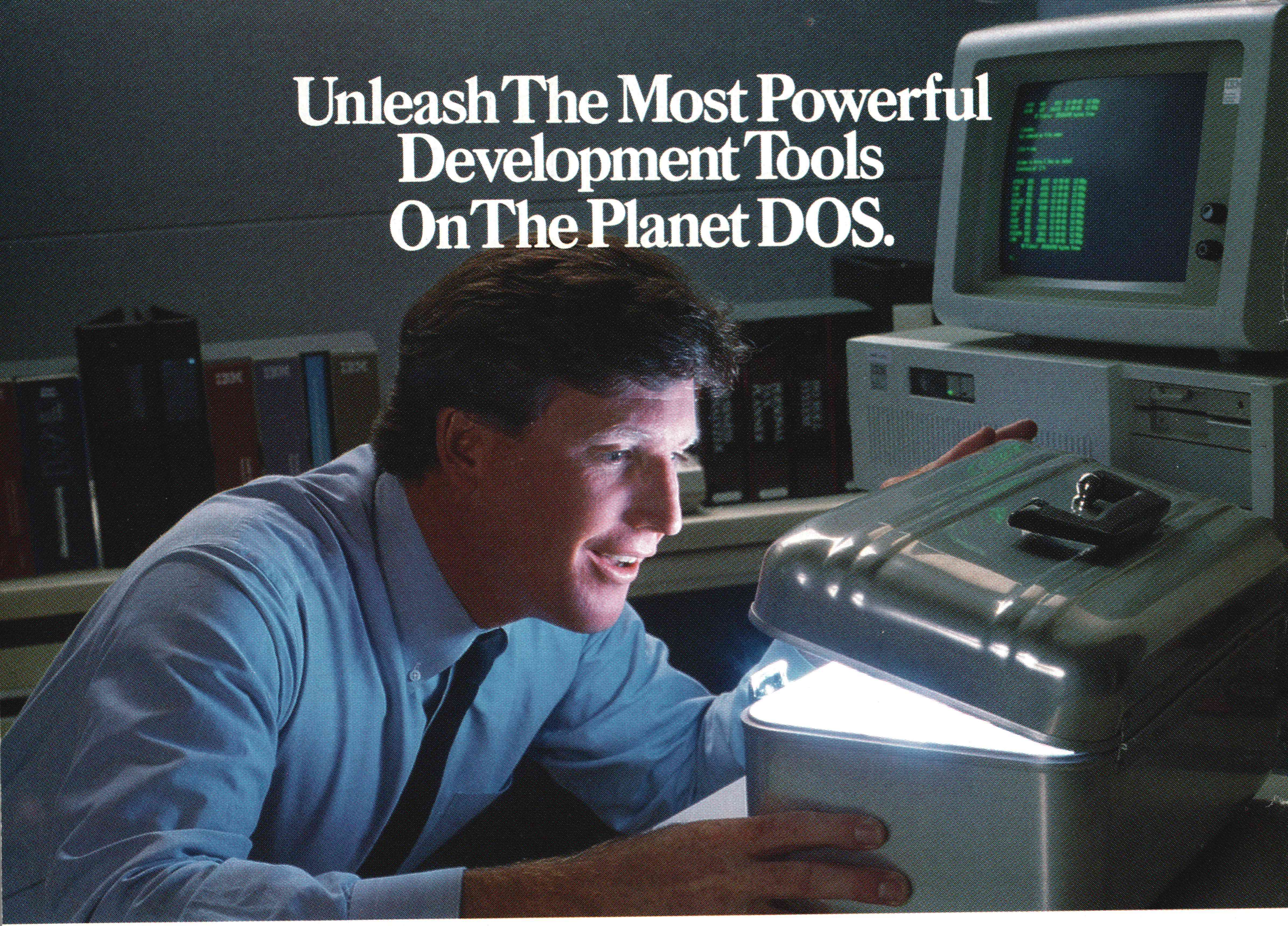
32000 Assembler

Comparing String Comparisons

Turbo Pascal Procedural Parameters

12
0 38351 16562

Unleash The Most Powerful Development Tools On The Planet DOS.



UNIFY DBMS/DOS. The UNIX World Leader Brings A New Dimension To DOS Application Development.

What happens as the DOS world expands? As a new generation of hardware takes over? As networking becomes more important? The potential is enormous. But until now, the tools to achieve it have been limited.

Now a leader from another world unleashes that potential: UNIFY® DBMS. The leading relational DBMS in the UNIX™ world. And now, the most advanced set of application development tools in the DOS world.

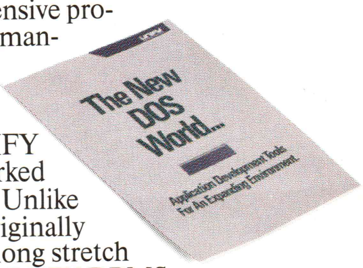
With UNIFY DBMS, DOS developers have new power to build more sophisticated applications than ever before possible.

The power to write high performance "C" programs that will access the data base, using Unify's Direct Host Language Interface.

The power of an industry standard query language—SQL.

The power of unmatched speed in production applications. Only UNIFY DBMS is specifically engineered for transaction throughput. With unique performance features like PathFinder™ Architecture multiple access methods, for the fastest possible data base access.

The power of comprehensive program development and screen management tools. Plus a state-of-the-art fourth generation report-writer.



What's more, with UNIFY DBMS, the potential of networked applications becomes a reality. Unlike DBMS systems which were originally single-user (and which have a long stretch to accommodate more users), UNIFY DBMS is a *proven* multi-user system.

And because UNIFY DBMS/DOS is the best of two worlds, it offers you the most powerful benefit of all: DBMS applications that can grow as your needs grow. From single user DOS. To networked DOS. To multi-user UNIX. All without changing your applications.

Call the Unify Information Hotline for our free booklet: The New DOS World. (503) 635-7777

UNIFY CORPORATION
4000 Kruse Way Place
Lake Oswego, OR 97034

Circle **no. 176** on reader service card.

The Peak of Performance
SCALE THE HEIGHTS OF PRODUCTIVITY
Sure, you've proven that in your hands a computer is a productive tool. But if you haven't teamed up with a SemiDisk you have heights yet to climb!
IT'S NO MERE RAMDISK
SemiDisk has been leading the way for Disk Emulators since their inception. If you've seen RAMdisks you know what it's like to load programs in an instant, and read or write files without delay. Unlike alternatives, the SemiDisk offers up to 8 megabytes of instant-access storage while leaving your computer's main memory free for what it does best - computing!
KEEP A GRIP ON DATA
Go ahead, turn off your computer. Take a vacation. With the battery backup option, your valuable data will be there in the morning even if you aren't. You'll sleep better knowing not even a 5 hour blackout will sabotage your files.
NEW LOWER SEMIDISK PRICES THAT WON'T SNOW YOU UNDER

	512K	2Mbyte
IBM PC, XT, AT	$495	$995
Epson QX-10	$595	$995
S-100, SemiDisk II	$799	$1295
S-100, SemiDisk I	$595	——
TRS-80 II, 12, 16	$695	$1295
Battery Backup Unit	$130	$130

Software drivers available for CP/M 80, MS-DOS, ZDOS, TurboDOS, and VALDOCS 2.
SEMIDISK
SemiDisk Systems, Inc.
P.O. Box GG, Beaverton, Oregon 97075
503-626-3104
Call 503-646-5510 for CBBS/NW, and 503-649-8327 for CBBS/Aloha, all SemiDisk equipped computer bulletin boards, (300/1200/2400 baud) SemiDisk, SemiSpool trademarks of SemiDisk Systems.
Circle no. 85 on reader service card.


Instant Replay
Has It Dawned On You Yet?
Prototype-->Program-->Tutorial
) Instant Replay - $ 89.95
Demo, Tutorial, and Prototype Generator. Create Replays of actual programs, or vapor ware only replays. Includes Screen Genie Screen Editor, Word Genie Text Editor, and a Keystroke Editor. Insert Pop-Ups, Prompts, Prototypes, and Users into the demo or tutorial.
) Assembler Genie $ 59.95
Assembly Language source code generator for including Screen Genie prototype screens into your actual programs. Supports almost every language.
) Plus Series - $ 59.95
Assembly Language Programming Tools for Turbo Pascal, MS Pascal, or C. Make your prototypes into real programs.
) NoBlink Accelerator - $ 49.95
Resident Cursor Enhancer. Choose your own cursor shape, color(non-blinking), or speed in any program. Uses only 6K of memory. Great time time saver. Why mouse around?
60 day satisfaction money back guarantee on all software. Not Copy Protected.
To order, or to obtain more information call or write
Nostradamus Inc. (801) 487-9662
3191 South Valley Street (suite 252)
Salt Lake City, Utah 84109
NOSTRADAMUS
Software Sourcery
Circle no. 251 on reader service card.
Visa, Amex, Master, Cod, PO, Check
For IBM PC, XT, AT and True Compatibles.
Outside US pays postage.

CONTENTS

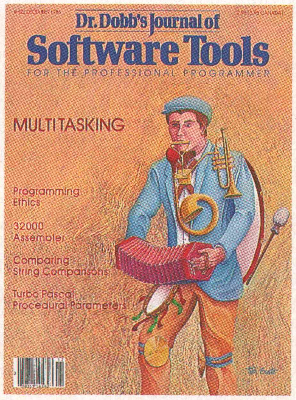
VOLUME 11, ISSUE 12

ARTICLES

COLUMNS

FORUM

PROGRAMMER'S SERVICES



About the Cover
A multitasking operating system is simultaneously bandleader and musician, and its results must be harmonious.

This Issue
With the advent of Intel's 80386 microprocessor, true multitasking is clearly visible on the DOS horizon. But the road to take us there—a multitasking operating system—is still being surveyed. Our feature article leads the way by showing how to build a task-scheduling kernel—the bedrock on which a multitasking operating system is constructed.

Next Issue
We'll begin the new year with a look at the present and future of the Motorola 68000 line of processors. How are the chips being used now, and what will the new members of the family be like? What will the 68030 (or rumored 68040) add in terms of speed and memory?

LANGUAGES THAT ARE CAUSING THE BIGGEST PROGRAMMING BACKLOG IN HISTORY ARE ALSO EATING NICE BIG HOLES IN OUR POCKETS.

Whether it's BASIC, COBOL, Pascal, "C", or a data base manager, you're being held back.

Held back because the language has frustrating limitations, and the programming environment isn't intuitive enough to keep track of what you're working on.

In the real world, there's pressure to do more impressive work, in less time, and for more clients.

We've been given some incredibly powerful hardware in recent times, but the languages aren't a whole lot better than they were 20 years ago.

So, whatever language you have chosen, by now you feel it's out to get you — because it is.

Sure, no language is perfect, but you have to wonder, "Am I getting all I deserve?"

And, like money, you'll never have enough.

Pretty dismal, huh?

We thought so, too.

So we did something about it.

We call it CLARION™.

You'll call it *"incredible."*

Distributed on 7 diskettes, CLARION consists of over 200,000 lines of code, taking 3+ years to hone to "world-class" performance.

With CLARION you can write, compile, run and debug complex applications in a New York afternoon.

Even if you're in Savannah.

It gives you the power and speed to create screens, windows and reports of such richness and clarity you would never attempt them with any other language.

Because *you* would have to write the code.

With CLARION you simply design the screens using our SCREENER utility and then CLARION writes the source code AND compiles it for you in seconds.

Likewise, you can use REPORTER to create reports.

Remember, only CLARION can recompile and display a screen or report layout for modification.

And with no time wasted.

All the power and facilities you need to write great programs, faster than you ever dreamed of.

Programs that are easy to use.

Programs that are a pleasure to write.

And to you that means true satisfaction.

You've coveted those nifty pop-up help windows some major applications feature. But you can't afford the time and energy it takes to write them into your programs.

That's the way it used to be.

So we fixed that, too.

CLARION's HELPER is an interactive utility that let's you design the most effective pop-up help screens that you can imagine. And they're "context sensitive," meaning you can have help for every field in your application.

Unlike the other micro languages, CLARION provides declarations, procedures, and functions to process dates, strings, screens, reports, indexed files, DOS files and memory tables.

Imagine making source program changes with the CLARION EDITOR. A single keystroke terminates the EDITOR, loads the COMPILER, compiles the program, loads the PROCESSOR and executes the program. It's that easy!

Our data management capabilities are phenomenal. CLARION files permit any number of composite keys which are updated dynamically.

A file may have as many keys as it needs. Each key may be composed of any fields in any order. And key files are updated whenever the value of the key changes.

Like SCREENER and REPORTER, CLARION's FILER utility also has a piece of the CLARION COMPILER. To create a new file, you name the Source Module. Then you name the Statement Label of a file structure within it.

FILER will also automatically rebuild existing files to match a changed file structure. It creates a new record for every existing record, copying the existing fields and initializing new ones.

Sounds pretty complicated, huh? Not with CLARION's documentation and on-line help screens. If you are currently competent in BASIC, Pascal or "C" you can be writing CLARION applications in a day. In two days you won't believe the eloquence of your CLARION programs.

Okay, now for the best part of all. You can say it in CLARION for $295.00—plus shipping and handling. All you need is an IBM® PC, XT, AT or true compatible, with 320 KB of memory, a hard disk drive, and a parallel port. And we'll allow a full 30 day evaluation period. If you're not satisfied with CLARION, simply return it in its original condition for a full refund.

If you're not quite ready to take advantage of this no-risk opportunity, ask for our detailed 16 page color brochure. It vividly illustrates the elegance of CLARION. Consider it a preview of programming in the fast lane.

Either way, the 800 call's a freebie.

SAY IT IN

CLARION™

A4ST/7

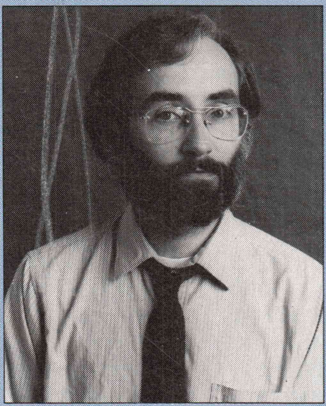
1-800-354-5444

VISA AMERICAN EXPRESS MasterCard

BARRINGTON SYSTEMS, INC. 150 EAST SAMPLE ROAD POMPANO BEACH, FLORIDA 33064 305/785-4555

Circle **no. 115** on reader service card.

EDITORIAL



It's two-thousand-odd miles from the East Coast of the United States to the West.

I know exactly in what year I learned this rough and homely statistic because is linked in my memory with a disturbing childhood insight into the nature of subjective time.

Halfway through the sixth grade, I realized that I had forgotten the fifth grade. Thinking back, I couldn't decide whether important events had occurred in the fourth grade or in the fifth or even in the early part of the sixth. Until then, each school year had seemed as distinct as a life unto itself, and summers were not vacations but a succession of heavens. But that winter the days and weeks began to slide into one another like segments of a telescope, and the only event in my life that I could with any certainty assign to my otherwise vanished fifth-grade year was reading that it was two-thousand-odd miles from the East Coast of the United States to the West.

By the early 1980s, decades were telescoping. While researching a book on what I thought was the first decade of the personal computer, I rediscovered an article written in the late 1970s by Sol Libes (rhymes with Phoebus) on the first decade of hobby computing. Sol referred to that article in an early issue of his magazine, *S-100 Microsystems*, in 1980. I read *S-100 Microsystems* scrupulously back then because I was working on S-100 systems and I enjoyed Sol's News and Views column. You'll find some of Sol's thoughts from that period in microcomputing history in our Archives section on page 8 (in case you're nostalgic for them, too).

Well, it was entertaining to read about the S-100 bus in 1980. The IEEE had just decided to establish a standard for the bus invented by Ed Roberts and MITS and seized upon by early microcomputer hobbyists and entrepreneurs. After Roberts denounced the other entrepreneurs as "parasites," entrepreneur Howard Fullmer renamed his company Parasitic Engineering, and the IEEE made Fullmer cochairman of the S-100 bus standard subcommittee. So, there were some differences of opinion.

Time continued to telescope, and within a few years the IEEE had adopted the S-100 bus standard and Sol's magazine had been purchased and promoted and put to death by a major publisher. Time had telescoped me into this job, and I wrote a brief epitaph for *Microsystems*.

Sol ignored all epitaphs and in the spring of 1985 began anew with *Micro/Systems Journal*, featuring most of the familiar contributors. There were changes that reflected changed realities, but it still had the quality one expected from Sol Libes.

Ahem.

This fall, M&T Publishing acquired *Micro/Systems Journal*. *Micro/Systems Journal* has joined the group of M&T publications that includes *Dr. Dobb's Journal of Software Tools*, *Turbo Tech Report*, and *Business Software Magazine*. Sol will remain editor and will remain in New Jersey, and he and I will consult with one another.

I'm looking forward to working with one of the legends of the industry. Of course, it's two-thousand-odd miles from the East Coast of the United States to the West. But we hope to telescope that distance.

Michael Swaine

Michael Swaine
editor-in-chief

Dr. Dobb's Journal of Software Tools (USPS 307690) is published monthly by M&T Publishing, Inc., 501 Galveston Dr., Redwood City, CA 94063; (415) 366-3600. Second-class postage paid at Redwood City and at additional entry points.

Article Submissions: Send manuscripts and disk (with article and listings) to the Assistant Editor.

Address Correction Requested: Postmaster: Send Form 3579 to *Dr. Dobb's Journal*, P.O. Box 27809, San Diego, CA 92128. **ISSN 0888-3076**

Customer Service: For subscription problems call: outside CA (800) 321-3333; within CA (619) 485-9623 or 566-6947. For book/software order problems call (415) 366-3600.

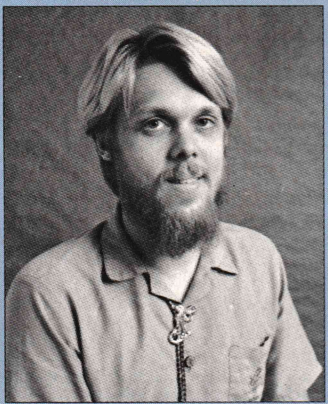
Subscription Rates: $29.97 per year. Outside U.S. add $45 airmail or $28 surface; Canada and Mexico add $27 airmail or $10 surface. Foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. For foreign subscriptions, TELEX: 752-351.

Foreign Newsstand Distributor: Worldwide Media Service, Inc., 386 Park Ave. South, New York, NY 10016; (212) 686-1520 TELEX: 620430 (WUI).

People's Computer Company

Dr. Dobb's Journal of Software Tools is published by M&T Publishing, Inc. under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a nonprofit corporation.

RUNNING LIGHT



Our latest issue of the listings on disk is out now. On it you'll find an eclectic collection of files, from Jan Steinman's Worm Memory Test for the 68000 to Richard Rodman's 32000 assembler (in this issue). You'll find McIvor's Radix Sort; Ashdown's Cubic Splines program; the benchmarks used in the Elkins/King turbo board review; a whole bunch of stuff from Holub's and Duncan's columns; and a couple of interesting Macintosh programs, including the Digital Dissolve demo by Mike Morton and Howard Katz's Mandelbrot program. See the *DDJ* ad catalog for more information.

This month's theme is operating systems, and we have a couple of interesting offerings. The lead article focuses on a task scheduling routine, the central kernel of any multitasking system.

Echelon's ZCPR is a Z80 CP/M work-alike that is not only more powerful than the original CP/M but also is more interesting: the code is tighter and takes good advantage of the Z80 instruction set. In this issue, Morris Simon takes a look at ZCPR.

This issue also features a complete 32000-family cross assembler by Richard Rodman. We're printing the listings in Cauzin Softstrip format. What do you think of the Softstrip listings? Please call or write.

Nick Turner
editor

STATEMENT OF OWNERSHIP, MANAGEMENT, AND CIRCULATION
(Act of August 12, 1970, Section 3685, Title 39, United States Code)

1. Title of Publication: Dr. Dobb's Journal of Software Tools, Publication No. 307690.
2. Date of Filing: October 6, 1986.
3. Frequency of Issue: (12 issues, $29.97).
4. Location of Known Office of Publication: 501 Galveston Dr., Redwood City, CA 94063.
5. Location of Headquarters of General Business Offices of the Publishers: 501 Galveston Dr., Redwood City, CA 94063.
6. Names and Addresses of Publisher, Editor, and Managing Editor: Publisher, Laird Foshay, 501 Galveston Dr., Redwood City, CA 94063. Editor, Michael Swaine, 501 Galveston Dr., Redwood City, CA 94063. Managing Editor, Vince Leone, 501 Galveston Dr., Redwood City, CA 94063.
7. Owner: M&T Publishing, 501 Galveston Dr., Redwood City, CA 94063.
8. Known Bondholders, Mortgages, and Other Security Holders Owning or Holding 1 Percent or More of Total Amount of Bonds, Mortgages or Other Securities: Markt & Technik, Hans-Pinsel-Strasse 2, 8013 Haar bei Munich W. Germany.
9. Extent and Nature of Circulation:
   - A. Total number of copies printed. Average number of copies each issue during the preceding 12 months: 59,297. Actual number of copies of single issue published nearest to filing date: 67,494.
   - B. Paid Circulation. 1. Sales through dealers and carriers, street vendors, and counter sales. Average number of copies each issue during preceding 12 months: 12,449. Actual number of copies of single issue published nearest to filing date: 16,530. 2. Mail subscriptions. Average number of copies each issue during preceding 12 months: 37,997. Actual number of copies of single issue published nearest to filing date: 40,519.
   - C. Total Paid Circulation. Average number of copies each issue during preceding 12 months: 50,446. Actual number of copies of single issue published nearest to filing date: 57,049.
   - D. Free distribution by mail, carrier, or other means, samples, complimentary, and other free copies. Average number of copies each issue during preceding 12 months: 2,104. Actual number of copies of single issue published nearest to filing date: 2,760.
   - E. Total distribution. Average number of copies each issue during preceding 12 months: 52,550. Actual number of copies of single issue published nearest to the filing date: 59,809.
   - F. Copies not distributed. 1. Office use, left over, unaccounted, spoiled after printing. Average number of copies each issue during preceding 12 months: 1,412. Actual number of copies of single issue published nearest to filing date: 600. 2. Returns from News Agents. Average number of copies each issue during preceding 12 months: 5,335. Actual number copies of single issue published nearest to filing date: 7,085.
   - G. Total. Average number of copies each issue during preceding 12 months: 59,297. Actual number of copies of single issue published nearest to filing date: 67,494.

I certify that the statements made by me above are correct and complete.

Laird Foshay, Publisher

ARCHIVES

"I am no businessman. Moreover, I am the kind of person who likes to take on a variety of projects and not tie myself to projects that take up all my time and run on for years. However, it has become apparent, after aproaching many people, that the only way an S-100 magazine will come into existence is if I assume the responsibility."—Sol Libes, *S-100 Microsystems*, vol.1, no.1, 1980.

"John C. Dvorak is publishing a monthly four page review of software. The primary emphasis is on North Star system software. The sample issue I received contained a great deal of useful information. John also distributes a wide selection of North Star software."—Sol Libes, *S-100 Microsystems*, Nov./Dec. 1980.

"I remember only five years ago buying my first copy of Basic for only $5. It was a mini-version of about 2K bytes of code on paper tape.The author sold it directly, as a hobby. He had a good full time job and this was just a little project on the side. The program provided only the very fundamental Basic functions...and there were a few bugs in the program.

"Today, things have changed radically. I have changed . . . from a pure hobbyist—to an entrepreneur that relies on his system for income. My needs have changed. I need powerful software which will allow me to do my tasks quickly. Also, I need software that is bug-free and does not consume my time with debugging.

". . . . I think that I am typical of most microcomputer users. I am sure that most of the computer hobbyists of two to five years ago are the entrepreneurs of today. It seems like all of my old computer hobbyist buddies are making money with their systems, one way or another. Although we still play around with our systems just for the fun of it, most of our time is now spent on income producing projects."—Sol Libes, *S-100 Microsystems*, Sept./Oct. 1980.

"The Altair-8800 used a 100-pin bus that was laid out by an anonymous draftsman who arbitrarily assigned signal names to groups of connector pins. Originally known as the 'Altair bus,' its name was quickly changed by other manufacturers of compatible products to the 'Altair/IMSAI bus' and the 'Altair/IMSAI/Protech bus.' This was too much, and at Atlantic City in 1976 Cromemco's Roger Melon coined the name 'S-100 bus,' which was universally adopted despite protests from MITS that it was still the 'Altair bus.' "—Sol Libes, *Microsystems*, May 1983.

Dr. Dobb's Journal of
COMPUTER
Calisthenics & Orthodontia
Running Light Without Overbyte

Large model C compiler performs like big-name brands includes free C tutorial

"As good as or better than most of the heavyweights..."
DR. DOBB'S, August 1986

The DATALIGHT C COMPILER is a full-implementation of the C language as defined in *The C Programming Language* by Kernighan and Ritchie. Supporting five memory models, DATALIGHT C has very fast compile, link, and execution times with a minimum of memory required. Our special introductory price is only $99 for the DEVELOPER'S KIT.

Optimize Your Code Generation

Now you can produce highly optimized code in the standard Intel object module format. The optimizations performed include common subexpression elimination, branch optimizations, constant folding, strength reductions, dead-code eliminations, and switch table compaction.

Five Memory Models Supported

DATALIGHT C provides five memory models so you can use the model that best suits your application.

Memory Models

Model	Code	Data
Compact	64k	total code & data
Small	64k	64k
Program	1M	64k
Data	64k	1M
Large	1M	1M

Compiling, One step...

Now with the one step DLC program you can compile, link, and create a .COM file. Also, one or more files can be compiled and linked using DLC.

Complete Library Includes Source Code

The UNIX compatible library includes complete source code. Experienced programmers can use the source code to configure and rebuild the library to suit the application.

Concise Documentation Included

The DATALIGHT C COMPILER and DEVELOPER'S KIT include a concise, to the point, programmer's manual. The 210-page manual is contained in an IBM-style three-ring binder which includes nine chapters, appendices, index, and easy to follow examples.

Datalight

BOX 82441
KENMORE, WA 98028
(206) 367-1803

30-DAY MONEY-BACK GUARANTEE

Try our DEVELOPER'S KIT for 30 days, and if you do not find it to be equal to, or better than, any compiler you have, or are using, then you may return it for a full refund. Also, if you return the compiler within 30 days, you may keep the C tutorial, a $39 value, for trying our DEVELOPER'S KIT.

Introductory Prices

DATALIGHT C without source $60
(compact & small memory model support)

DEVELOPER'S KIT with source $99
(five memory models supported)

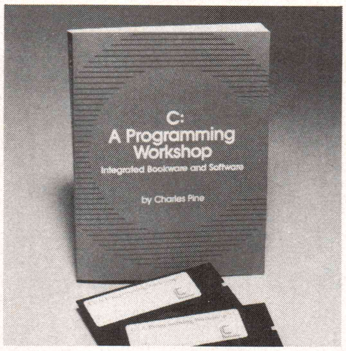
Not Copy Protected

ORDER TOLL-FREE TODAY!

1-(800) 221-6630

*SPECIAL OFFER!

ORDER the DEVELOPER'S KIT by December 31, 1986 and receive FREE...



"Combines print and software technology to create the most integrated new type of training system I have ever seen."

ADAM GREEN, INFO WORLD
January 27, 1986

"C:A Programming Workshop" is a complete C language learning package with an integrated compiler and screen editor. Learn the C language with tutorials, quizzes, and program exercises.

Add $5 for shipping in US/$15 outside US
COD (add $2.50)

*Limited offer available exclusively to readers who purchase directly from DATALIGHT.

Magazine Reviewers Shocked by DATALIGHT'S PERFORMANCE...

"Reviewing this compiler was quite a surprise for us. For such a low price, we were expecting a "lightweight" compiler. What we got was a package that is as good as or better than most of the "heavyweights." DATALIGHT C implements a complete C language. It also compiles quickly, doesn't take up much disk space, and looks impressive in the benchmarks."

DR. DOBB'S, August 1986

"This is a sharp compiler! ...what is impressive is that DATALIGHT not only stole the compile time show completely, but had the fastest Fibonacci executable time and had excellent object file sizes to boot!"

Chris Skelly, COMPUTER LANGUAGE
February 1986

DEVELOPER'S KIT (VERSION 2.12)

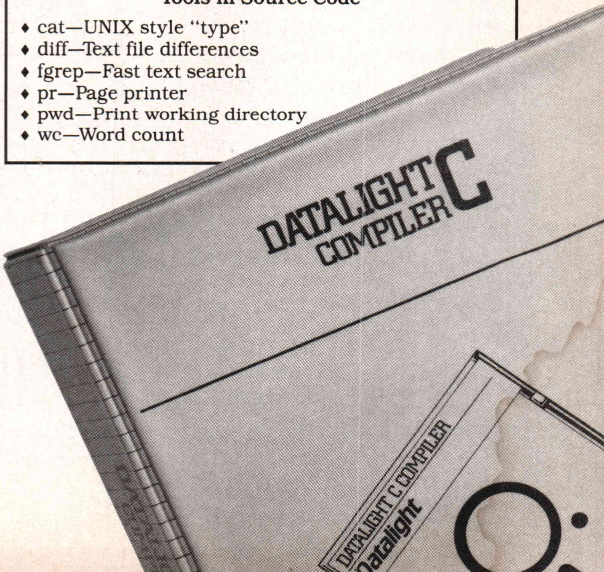
- Full UNIX System 5 C language plus ANSI extensions.
- Fast/tight code via powerful optimizations including common sub-expression elimination.
- DLC one step compile/link program.
- Multiple memory model support.
- UNIX compatible library with PC functions.
- Compatible with DOS linker and assembler.
- Third-party library support.
- Automatic generation of .COM files.
- Supports DOS pathnames, wild cards, and Input/Output redirection.
- Compatible with Lattice C version 2.x.
- Interrupt handling in C.
- Debugger support.
- ROMable code support/start-up source.

MAKE Maintenance Utility

- Macro definition support.
- MS-DOS internal commands.
- Inference rule support.
- TOUCH date manager.

Tools in Source Code

- cat—UNIX style "type"
- diff—Text file differences
- fgrep—Fast text search
- pr—Page printer
- pwd—Print working directory
- wc—Word count



Circle **no. 203** on reader service card.

LETTERS

Burning C

Dear *DDJ*,
So, the high priest of obsCurity defends the purity of the faith against the unwashed. Or was Allen Holub's August Viewpoint merely a stalking-horse? Better make that a dead horse; let's flog it some more.

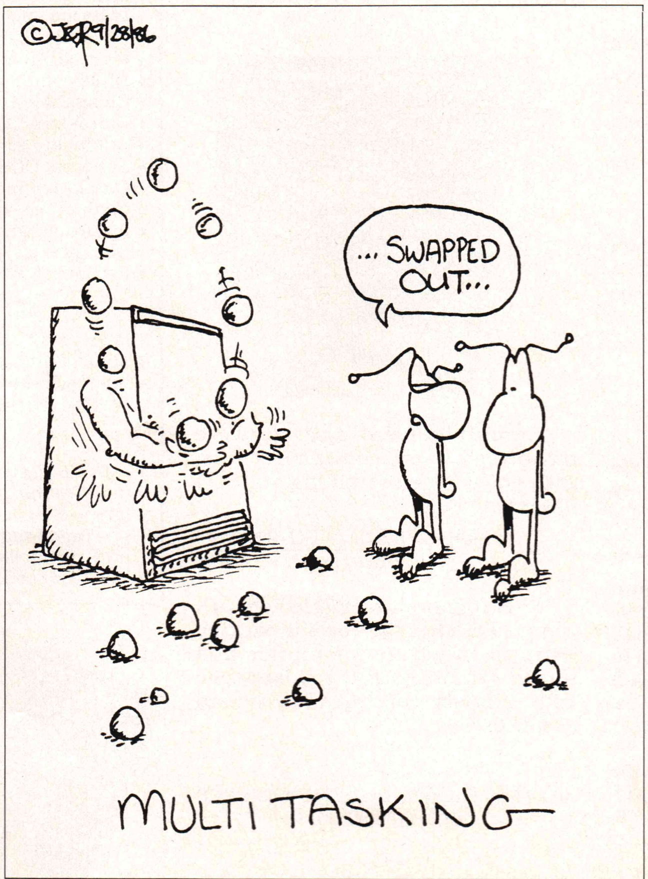
First of all, my programming credentials are better left unsaid, though I do program regularly in two assembly languages, read (manual in hand) two or three more, own and try to read Knuth, hate strongly typed languages, dislike long-winded compilers, and love Forth. And as for C, it is everything negative that has been published in *DDJ* et al. You might accurately surmise that I wouldn't want to take Mr. Holub's C classes and that U.C. Berkeley wouldn't let me anyway. But. . . .

It doesn't take too long for the non-C-fluent to realize what is the problem with the language and their grasp of it: that the ordinary control structures they use in any other language are damned difficult to pick out of a C listing. They are so difficult that someone has even written a book about it: *The C Puzzle Book*. In spite of Mr. Holub's wishy "C is difficult [because of] the ways that pointers are used and so forth" and washy "assembly-language programmers usually have little difficulty learning about pointers," it is the symbols of the language that boggle—you are continually reminded that everything you know is wrong. *% != || %s\n* indeed!

```
main( )
  {curse
    ( K = =~R && ^s %
                   >> C )
      = "@#%+# K&R's C";
  }
```

Combine two glaring stupidities: a language-wide propensity for the least possible typing (these guys are two-fingered for sure) and a finicky syntax based upon *{ } ( ) ; /* */*. (God forbid that you miss one.) Add a heavy dose of self-righteous disdain for anyone less than an operating system hacker; a black-box ("well, this bit works so let's forget comments") library archive; vested interests of book publishers to sell books that almost explain; and finally, those software firms and authors who already have "it" on one machine (so let's port it everywhere). Result: a popular language with commercial interests that's difficult to learn yet easy to use and that pays increasing dividends to those who have the most time and code invested.

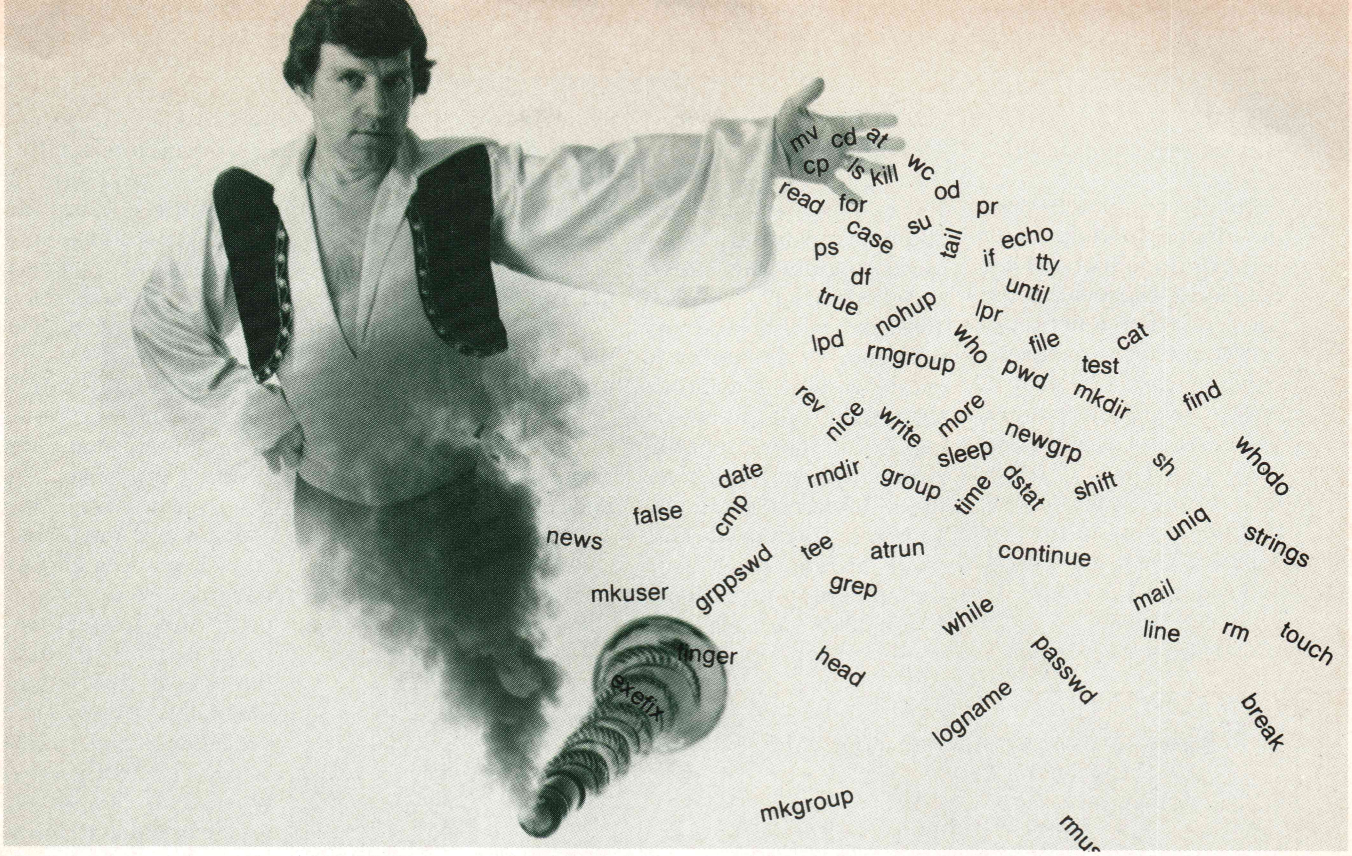
Anyway, "Easy C" by Orlin and Heath did one thing that all of C Chest never has: permitted me to read and use C within an hour. The authors' approach to C is unique in that they didn't say everything I knew was wrong. So, with some quick typing, I've my own EASYC.H. Interestingly, mine isn't much like Orlin and Heath's but more like Forth (the control constructs) and 9900 assembly language (the operators). I use it to write code that I can read tomorrow, and with its listing in hand, I can decipher even Mr. Holub's constructs.



I once wrote a disassembler that printed out the English translation of instructions—for example, *Load Immediate* for *LI*. After a while I got impatient with its long-windedness and put the "correct" ones into the data. I guess I learned the language. EASYC.H works in the same way. After a while, *INC* gets more tiresome to type than ++. The neat thing about EASYC.H is that I can mix the representations within the same program. Allen Holub reveals more the expert's impatience than insight into how Easy C can aid the beginner.

And yes, the defines for >=, <=, >, < are dumb. But the ones for *&&, ||, !, ==, &, ^, |, ~, <<, >>, ++, --,* and *%,* are godsends for someone who can't stand to look at (or use) C more than once a month. If Allen Holub wanted to do something really useful with C Chest, he might publish yet another pipeline program that would *purify yours.c his.c* and its converse *bastardize his.c readable.c*.

Purify might be already done by the first pass through the parser, but I could see using *bastardize* every time I downloaded a C program. Or typed in one of Mr. Holub's gems of conciseness. That'll work very nicely—it will require the least typing and provide the most readability.



Your wish is our commands.

UNIX operating systems have a well deserved reputation for their powerful command set. Now PCNX gives you the same powerful commands. And a lot more.

Built with Wendin's Operating System Toolbox, PCNX is the only multitasking, multiuser operating system that can put the popular features of UNIX on your personal computer (IBM PC, AT, XT, or true compatible) for under $100.

If you're already familiar with the UNIX environment you know what that means. If you aren't, PCNX is your chance to discover what you've been missing at a price you can finally afford.

Designed specifically for systems programmers, PCNX has all the features that have made UNIX the preferred environment for systems development. Commands that let you perform complex operations with a few keystrokes. Plus syntax that lets you compile programs in the background, handle piping and I/O redirection, and build shell script files.

And that's not all.

PCNX links the popular Bourne shell to Wendin's unique kernel. As a result, you have access to more than eighty enhanced system services built into all our Personal Operating Systems.

Like our other Personal Operating Systems, PCNX also runs well-behaved MS-DOS programs, is fully compatible with the MS-DOS file system, and uses existing MS-DOS compilers, linkers and utilities.

In addition, it comes with complete source code that lets you see for yourself exactly how the system works.

So if you've ever wished you had the power and flexibility of UNIX on your PC, why not stop wishing and order PCNX today.

PCNX. From Wendin. Only $99.

Ask about our other products for the IBM PC and true compatibles.

PCVMS™
Multitasking, multiuser version of DEC's powerful VAX/VMS operating system. Runs most MS-DOS programs.

XTC®
The ultimate programmer's editor. Multitasking macro language plus multiple linkable windows and buffers.

OPERATING SYSTEM TOOLBOX™
Complete software construction set that lets you build your own multitasking, multiuser operating systems.

All products priced at $99 with source code included.

DEALER INQUIRIES WELCOME

Foreign orders inquire about shipping.
Domestic orders add $5.00/1st item, $1.00 each additional item for shipping, handling, and insurance. We accept Visa/MC, American Express, COD, and Bank Drafts drawn on U.S. Banks.
Washington residents add 7.8% sales tax.

MS is a trademark of Microsoft, PC-DOS is a trademark of IBM. UNIX is a trademark of AT&T. VAX/VMS is a registered trademark of Digital Equipment Corporation.

Wendin and XTC are registered trademarks of Wendin, Inc. PCNX, PCVMS, Operating System Toolbox, and Personal Operating System are trademarks of Wendin, Inc.

WENDIN®
BOX 266
CHENEY, WA 99004

The people who make quality software tools affordable.

ORDER HOTLINE
(509) 235-8088
(MON.-FRI., 8-5 PACIFIC TIME)

Circle **no. 112** on reader service card.

LETTERS
(continued from page 10)

Finally, computers were developed to help math wizards do millions of calculations—big deal. I use mine to write letters, call up CIS, and draw pictures. I couldn't care less why C was put on this earth and will resist the fascist Pascal to my dying day. If you're new to the programming business, you should be learning Forth, not Pascal.

Frederick Hawkins
1020 N. 6th St.
Allentown, PA 18102

Structured Programming

Dear *DDJ*,

I was surprised that Mr. Shammas' column on generic routines (Structured Programming, August 1986) did not include any reference to the ability to create generic C routines through the preprocessor. I should point out that this is not my idea. The technique is described in *Advanced C: Food for the Educated Palate*, by Narain Gehani (Computer Science Press, 1985)—a useful and well-written book. I include a generic sort routine [see Table 1, below] that I have found useful for sorting arrays of various types. It can be instantiated by:

```
GENERIC_SORT(int_sort,
                    int, >),
```

or

```
GENERIC_SORT-
        (float_sort, float, <)
```

The sort I used is the standard Shell sort, à la K & R.

Keep up the good work!

Kit Kauffmann
Algorithms Unlimited
P.O. Box 3516
Ogden, UT 84409

```
#define GENERIC_SORT(NAME, ELEM_TYPE, OP) NAME(v, n)\
        ELEM_TYPE v[ ]; int n;\
        {\
                ELEM_TYPE temp;\
                int gap, i, j;\
                for(gap = n/2; gap > 0; gap /= 2)\
                        for(i = gap; i < n; i++)\
                                for(j = i-gap; j >= 0 && v[j] OP v[j+gap];j -= gap)\
                                {\
                                        temp = v[j];\
                                        v[j] = v[j+gap];\
                                        v[j+gap] = temp;\
                                }\
        }
```

Table 1: *Generic sort routine in C*

```
100  REMark Example of a generic Shell sort in Sinclair SuperBasic
110  :
120  DEFine PROCedure shell_sort(L,Num)
130    LOCal Offset%,I,J,K, Divide_And_Conquer,Get_In_Order,In_Order,Temp$
140    Offset% = Num
150    REPeat Divide_And_Conquer
160      IF Offset% <= 1 THEN EXIT Divide_And_Conquer
170      Offset% = Offset% / 2
180      REPeat Get_In_Order
190        In_Order = -1
200        K = Num - Offset%
210        FOR J = 0 to K
220          I = J + Offset%
230          IF L(J) > L(I) THEN
240            In_Order = 0
250            Temp$ = L(I)
260            L(I) = L(J)
270            L(J) = Temp$
280          END IF
290        END FOR J
300        IF In_Order THEN EXIT Get_In_Order
310      END REPeat Get_In_Order
320    END REPeat Divide_And_Conquer
330  END DEFine shell_sort
```

Table 2: *Generic Shell sort in Sinclair SuperBasic*

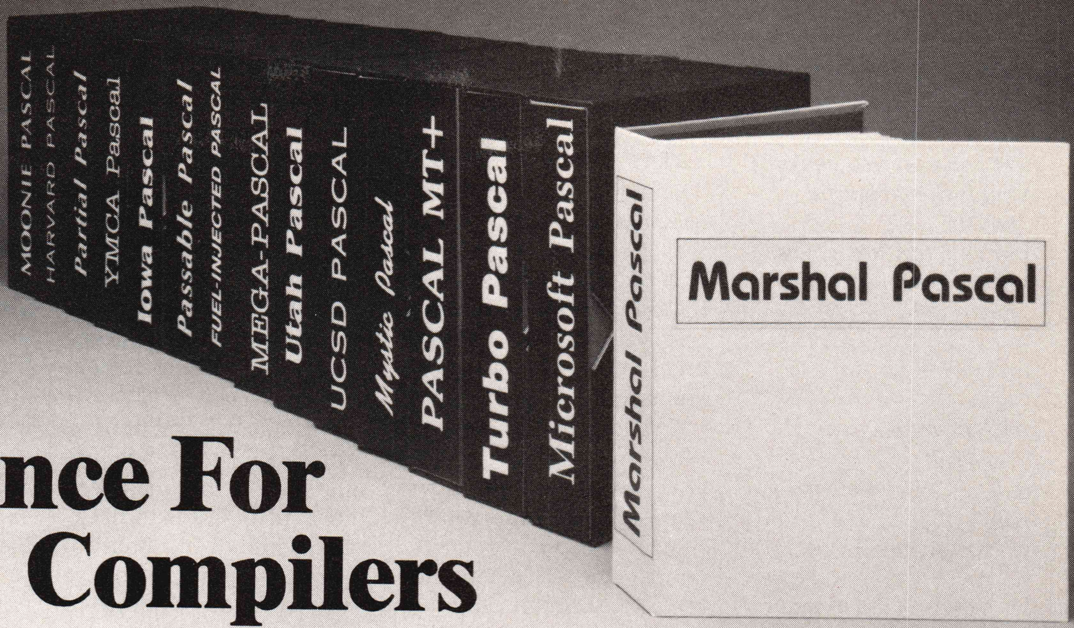
Dear *DDJ*,

I read with interest the August 1986 Structured Programming column about generic procedures in Ada and Modula-2. I agree that generic procedures are powerful and desirable features.

Both languages implement generic procedures in an overly complicated manner, however. I use a language in which all procedures and functions are generic because all formal parameters are typeless until the procedure or function is called. Another feature of this language that helps toward generic procedures and functions is coercion between data types. You can set string variables equal to numeric types and vice versa. Integer variables set equal to a real number will automatically round the real number to the nearest integer. Real number variables can take on the value of integers. I feel that late binding procedures and functions and coercion between data types provide an ideal programming environment. The language that provides these features is Sinclair SuperBasic as implemented on the Sinclair QL. This version of BASIC is highly structured and flexible, supporting such features as bitwise manipulation. I have included a listing [Table 2, left] of a generic Shell sort in SuperBasic that can sort strings, reals, and integers. Love your magazine!

Michael A. McCoy
Box 84 King Edward's Ct.
Rocky Mount, NC 27803

DDJ



A New Reference For Pascal Compilers

	ACKERMAN		SIEVE		I/O		GAUSS-SEIDEL		FLOATING-POINT	
MARSHAL PASCAL	11.9	5.1k	4.8	3.4k	1.9	6.8k	5.7	11.5k	10.5	5.3k
IBM PASCAL	12.4	34.7k	11.7	27k	2.6	28k	7.6	34.6k	10.5	21k
TURBO PASCAL	22.7	11.6k	14.2	11.5k	2.2	12.5k	4.7	13.5k	28	11.4k
MICROSOFT C 4.0	15.9	9.3k	5.8	6.5k	1.9	8.9k	6.0	23.6k	33	19.6k
	Time (sec.)	Code size	Time (sec.)	Code size	Time (min.)	Code size	Time (sec.)	Code size	Time (sec.)	Code size

Unparalleled Speed and Power

Marshal Pascal™ is the most highly code-optimized Pascal compiler for PCs. Period. In fact, Marshal Pascal produces code so fast and compact that even the most efficient C compilers fall behind in performing many operations. Also, Marshall Pascal is an ISO implementation thus offering portability to other computer environments. You may address as much memory as your operating system allows and a variety of memory models are supported. Among the useful extensions included are: separate compilation of modules (both Modula-2 and Pascal forms), structured constants and structured function values, variable-length string types and procedural parameters.

Turbo Pascal® Dialect Filter

Our Dialect Filter helps bring your Turbo Pascal programs over to an ISO/Marshal-readable format. Watch your present Turbo programs run in a fraction of the time and code space!

8087-80287 Support

Marshall Pascal supports the Intel 8087-80287® math processors *inline*. If the user has no math chip then '87-287 code emulation is a provided option.

Microsoft Linkability

Marshal Pascal's true relocatable linker allows the user access to the Microsoft family of languages and assemblers. A flexible object code librarian and powerful overlay capabilities are also included.

Powerful Compile Options

Marshal Pascal permits a number of compile options including an optimization by-pass for speedier compiles, I/O "fine-tuning", constant "folds" and a syntax evaluator just to name a few. A wealth of compile-time checks permit the user to find the more subtle logic errors reducing debugging time enormously.

Space does not permit us to list all the features of Marshal Pascal, all of which are provided USER options, not additional cost options. Marshal Pascal is not broken apart to make many products out of what should be only one good one.

The Price?

$189.

includes everything.

Supports PC-DOS®, MS-DOS®, CP/M-86® and Concurrent DOS®.

To order your copy of **Marshal Pascal,** call: **(800) 826-2222**
In California: **(415) 947-1000**

Marshal Language Systems – 1136 Saranap Ave. – Walnut Creek, CA 94595

VIEWPOINT

And you may ask yourself
What is that beautiful house?
And you may ask yourself
Where does that highway go to?
And you may ask yourself
Am I right? . . . Am I wrong?
And you may say to yourself
My God! What have I done?
from "Once in a Lifetime"
—David Byrne, Brian Eno

I've a confession to make—I'm really a composer, a musician, not a computer programmer at all. I went back to school to learn how to build synthesizers, and learning about computers was a side effect. I like to program and to design hardware, but I've become increasingly uncomfortable about engineering as a profession. More accurately, I have reservations about the uses to which my work can be put.

My main reservations have to do with weapons construction. I don't like it, but it can be difficult to get away from. After graduating, I worked on miltary-related projects for a while without really worrying about the ultimate purposes of my work. The work was technically interesting and wasn't closely related to weapons development—or so I thought. One day, after working for a year on a consulting project—building a manufacturing control system for airplanes—I discovered that the airplanes in questions were F-16s. This project was much too close to weapons production for me to be comfortable; I could no longer ignore what I was doing.

by Allen Holub

This put me in a real quandary. On one hand I was disgusted by what I was building; on the other hand I had committed myself to finishing the project and I'm very old-fashioned about fulfilling promises. If I just left the project, it wouldn't get finished (I was managing the software development and had written a good portion of the code). I dealt with my conscience by doubling my consulting rate and giving the extra money away to peace groups. The client was not happy but had no choice. I finished up the project as quickly as I could and left.

I had no trouble finding another consulting job. My new client's reaction to what had transpired at the old job was more amusement than anything else, and I thought I had found a company with a conscience. Unfortunately, the new company started producing products that were sold almost exclusively to defense contractors. I wasn't working on these, but I was still uncomfortable. Giving money to peace groups didn't satisfy my conscience anymore.

There is no easy way for me to deal with conflicts between my professional interests, financial needs, and conscience. The problems have been made worse by the Reagan administration's emphasis on defense spending—let's face it, an ever increasing percentage of the interesting and well-paying jobs are in weapons work. My solution has been to move gradually out of the field. I only take consulting projects that I'm convinced are not related to weapons and with companies that don't do weapons work. I'm a DUMPY, a downwardly mobile young professional. My income has diminished, but it's still adequate. More important, I'm doing things that I think benefit society.

To make my "sudden onslaught of conscience" (to quote one of my managers) more intelligible, I'll explain some of my political views. I've come to believe that the out-of-control arms buildup is not just the responsibility of government. Nuclear weapons don't spring, like Athena, fully armed from the head of Zeus. They're designed and built by engineers and programmers, by you and me. If we refused to build them, they wouldn't exist.

How is it, then, that a programmer who supports a nuclear freeze goes off to work every morning for a defense subcontractor that's making control systems for F-16s? How can we allow a regular paycheck to be more important than the ultimate harm we are doing to society? Weapons work is grave digging, quite literally. What good is a nice fat paycheck when your life has been reduced to a heap of radioactive slag? If nuclear weapons are used again, won't the developers be just a much mass murderers as the madmen who pushed the buttons?

Of course, there are people who have thought about these issues and have come to the opposite conclusion from mine: They believe that a strong nuclear arsenal makes the world a safer place. Though I think they're wrong, these people are at least acting according to their beliefs. I have a hard time, though, with people who refuse to look at the issues and build weapons anyway. We don't mitigate the effect of our work by not analyzing what we're doing. That the human race could be put to an end, and that some humans seem to be trying to do just that, should give any thinking person pause. That people can just sit back and let this happen is unconscionable. That some engineers claim to oppose the use of nuclear weapons or don't look at the issues of weapons production and still help to build them is worse.

What I'm trying to do is to convince you to do something. Those who don't like the work their companies are doing could try to change things. If they can't, they could get another job—like Peter Hagelstein, the key Star Wars scientist who recently resigned from Lawrence Livermore Lab because he was reportedly so unhappy about doing weapons research with X-ray lasers.

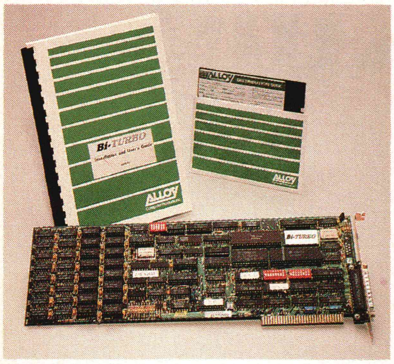
I strongly believe that an engineering ethics course should be a required part of every EECS curriculum—a course that would get people to look beyond that fat paycheck to the effects of their work on society. Those of us who work for universities could try to get such a course going. We can contact our legislators. We can work with peace groups. Whatever we do, we musn't sit around and wait for the bombs to be dropped—or they will be.

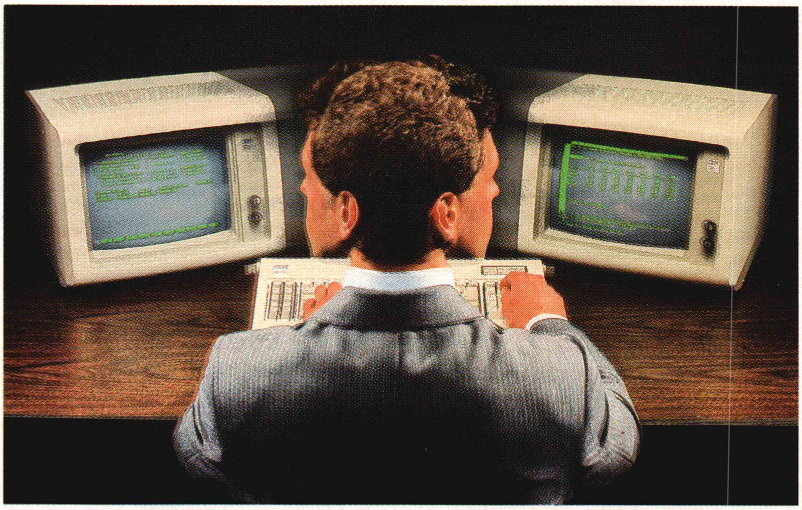
DDJ

Vote for your favorite feature/article.
Circle Reader Service **No. 1**.

When You Need To Do Two Things At Once —

Bi-TURBO™ . . . from ALLOY





Alloy's Bi-TURBO combines the features of today's best-selling boards and adds distinctive Alloy engineering experience to create the first accelerated dual-tasking card for personal computers. Bi-TURBO is the *complete* dual-tasking solution, with everything you need to be twice as productive.

- Adds power to PCs, XTs and even ATs
- Adds a full 640 KB of "second task" RAM, with its own 8 MHz processor
- Adds 256 KB of RAM-based disk cache to speed disk access for both tasks
- Easy-to-use dual tasking software — switch *full screen* windows with a keystroke!
- Print spooler, with *multi-printer* capability

With One PC and Bi-TURBO, Do the Work of Two PCs

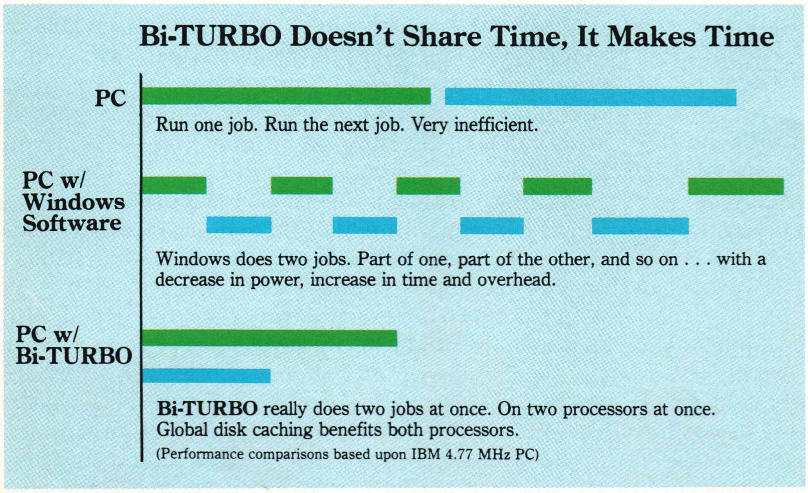
You're in the middle of a three-hour mail merge and you need to run a spreadsheet — NOW! Bi-TURBO to the rescue! Press one key on your keyboard and you're running the spreadsheet — while the mail merge task continues to run *at full speed.*

You get both jobs done, and Bi-TURBO's RAM-based disk caching actually SPEEDS UP the mail merge!

$995! — Bi-TURBO Saves Money and Time.

Bi-TURBO gives you the features and benefits of multi-function cards and windowing software — *and much more performance* — for less.

Bi-TURBO is the *complete* power upgrade for your XT or AT, with added 8 MHz processing power, added RAM, disk caching, print spooling, truly *easy-to-use* dual-tasking software and optional 8 MHz 8087 math co-processor.



Call your local distributor today . . . and Bi-TURBO.

Crystal Computers, Inc.
Lenaxa, KS 66214
(913) 541-1711
Irving, TX 75063
(214) 929-1300

FA Components
Elmhurst, NY 11373
(718) 507-1444
Ft. Wayne, IN 46808
(219) 432-8540
Greenville, SC 29607
(803) 288-2422

First Source Distributing
Salt Lake City, UT 84119
(801) 973-0074

Micro Computer Distributors, Inc.
Huntington Beach, CA 92649
(714) 895-5841

Micro Wholesalers, Inc.
Hunt Valley, MD 21030
(301) 666-5300

PC Distributing, Inc.
Northbrook, IL 60002
(312) 298-1400

PGI Corporation
Tempe, AZ 85281
(602) 967-1421

Super Source
Norcross, GA 30071
(404) 441-3451

Vitek
San Marcos, CA 92069
(619) 744-8305
San Jose, CA 95131
(408) 436-8026

Vitronix Corporation
Westboro, MA 01581
(617) 366-1144

W4 Micro Distributors
Birmingham, AL 35209
(205) 945-8310

Western Micro Systems
Scottsdale, AZ 85260
(602) 948-4240
Mountain View, CA 94043
(415) 964-2050
Northridge, CA 91324
(818) 700-9922
Redmond, WA 98052
(206) 881-6737

ALLOY
Computer Products, Inc.

Alloy Computer Products, Inc., 100 Pennsylvania Avenue, Framingham, Massachusetts 01701. (617) 875-6100, TWX: 710-346-0394
Alloy Computer Products, Inc., 9 Executive Circle, Suite 240, Irvine, California 92714. (714) 261-7661
In Europe: Alloy Computer Products (Europe) Ltd., Cirencester, Gloucestershire, England. Tel. 0285-69571, Tlx: 43340
In Australia: Alloy Australia Pty. Ltd., Suite 3, 318-322 Stephensons Road, Mount Waverley, Victoria, Australia 3149. Tel. 011-613-277-1333

Circle no. **178** on reader service card.

A Multitasking Kernel

by Ken Berry

Tele implements an 80386-like task scheduler on 'unsecure' processors.

Tele is a multitasking operating system for the IBM PC and compatibles that is written in C and assembly language. This article describes in detail the most critical part of Tele, the task scheduler, which handles the allocation and distribution of CPU time to the various tasks in the system. If you roll your own operating system, the task scheduler probably will present the most difficult problems. I hope to provide enough information to make it possible for you to design a task-scheduling kernel for your own multitasking operating system.

Tele contains three large blocks of code: the run-time library, the file system, and the task scheduler. These account, roughly, for 1/3, 1/2, and 1/6 of the total amount of code.

The Run-Time Library

The Tele run-time library has unique features, but all such libraries are essentially alike. They encapsulate algorithms common to a wide range of applications and thereby reduce the amount of time needed to code application programs.

The File System

Tele's file system is complex because of my requirement that it support both MS-DOS and Unix media. All my development tools were written for MS-DOS, so all the files had to be on MS-DOS media. I could therefore use MS-DOS for the file system as I implemented other parts of Tele. I eventually completed the Tele file system to significantly improve performance, but everything else was fully tested at that time.

Task Scheduler

The security kernel, or task scheduler, of an operating system is responsible for the allocation of system resources to tasks. It takes its name from military and financial applications in which security means protection against intrusion and sabotage. I don't worry about that kind of security because I don't leave my computer connected to the public communications system. I do, however, make programming mistakes. In particular, I sometimes mess up the stack and cause a program to return to the wrong location. When a program runs wild, it can execute any conceivable code, so there is no logical difference between a saboteur program introduced by a spy and a benign program that has run wild. An operating system that cannot be crashed by an application is identical to one that is secure from intrusion.

In order to make a secure (or crashproof) system, you must have a task scheduler that enforces certain rules on every other program. The rules determine what data can be accessed and where control can be passed; they must be enforced on the execution of every instruction. Processors such as the 80286 and 80386 check every instruction for security violations. Other processors, such as the 68000 and 32000 series can also support secure systems. The more common 8086 and 8088 processors cannot.

The necessary, and sufficient, rules for a secure system are simple. All programs are ranked according to security level, or degree of trust. A task can access data only at its own level or at a less trusted level. It can call another program or subroutine at its own level or at a more trust-

Ken Berry, P.O. Box 966, Jackson, CA 95642-0966.

ed level. Conversely, no program can access data from a more trusted level, nor can control be passed to a less trusted program in a call or branch. On a secure processor the task scheduler enforces these rules by loading the protection registers with appropriate values.

Tele implements a task scheduler on unsecure processors like the 8080 and 8086 so that it presents the same environment regardless of processor. Eventually the bugs that result in programs running wild will be found and corrected. Provided that the program is not a saboteur, it will then run exactly the same on any processor supported by Tele.

A Detailed Description

The listings that accompany this article implement the task-scheduling kernel. You can enhance your applications by incorporating these functions into MS-DOS. An initiation function alters the 8086 interrupt vector and reprograms the system clock to support Tele. This is done without damaging MS-DOS, which remains in memory and is available for use as a file system and other purposes. A termination function removes Tele and restores MS-DOS completely.

Function *t__krnl* in Listing One, page 50, is the security kernel. This program is an infinite loop. It determines the next task to be executed and passes control to it. The task will eventually be interrupted by the system tick clock and control will return to *t__krnl*. The terminated task is then stored in an appropriate queue, and the loop repeats.

Listings Two through Four are presented as supplements to Listing One. Listing Two, page 58, contains several macros used with the assembly-language functions; Listing Three, page 61, contains various assembly-language subroutines that are intended to support Listing One; and Listing Four, page 66, contains additional assembly-language programs that deal with timing concerns.

Scheduling

Tele uses a scheduling algorithm that supports real-time processing as well as preemptive multitasking. Much of the system code executes as independent tasks. Also, the hardware device drivers are fully interrupt-driven. These features make the task scheduler and its associated functions intricate. The end result is efficient processing of several types of tasks.

Figure 1, page 18, illustrates the scheduling queues manipulated by the task scheduler. The current state of a task is equivalent to the queue in which it resides. The execution queue (X) contains tasks ready for execution. The wait queue (W) contains tasks waiting for a particular (real) time to execute. The priority queue (P) is actually a group of queues used to sort tasks by priority. Also shown in Figure 1 is the pointer to the currently executing task (C). The listings contain a symbol, *_tsk*, that locates the control table for the current task.

All tasks are represented by a *t_task* structure, which is defined in Listings One and Two. The *t_ftask* structure contains *t_task* as well as several additional fields. Application tasks are represented by *t_ftasks*; they may have private memory areas and files assigned to them. Internal system tasks only use *t_task* to conserve space—they all share memory and files private to the system.

A task is said to be in a particular queue if the linkage pointers in the task's *t_task* structure are linked in the queue. Therefore managing tasks amounts to manipulating the linkage pointers in the *t_task* structure. The security kernel is a state machine. It implements rules for changing the states of tasks by changing the queue linkages.

(continued from page 17)

Tele uses a fixed execution interval—that is, a periodic interrupt is used to divide time into equal intervals. Each interrupt marks the end of an application task execution interval. Higher priority tasks are allowed to execute more often, but all tasks are equal in being allowed to execute for one interval at a time. Another way of implementing priorities is to vary the execution interval, allowing higher priority tasks to execute for longer each time. I had to use the fixed-period method because a standard PC has only one programmable interrupt available.

The interrupt is generated by an 8253 clock/timer circuit attached to an 8259 interrupt controller. The 8253 provides three programmable timers: one of these is used to refresh memory (Tele makes use of this for high-resolution timing, as explained later), another generates the signal used for the operator alarm speaker, and the third timer can be used to interrupt the processor.

Tele actually requires three interrupts. One is necessary to measure time and maintain a time-of-day clock. Another is needed to update the console display. The display driver Tele uses is six times more efficient than the standard BIOS, besides supporting window overlays. The display driver is relevant here only because it requires a periodic interrupt synchronized to the display hardware. The third interrupt required is to preempt application tasks. Fortunately, it is possible to serve all three functions with a single interrupt provided it has a fixed period (which must be harmonically related to the display hardware).

Function *t_tick* in Listing Four services the system tick interrupt. Because bad things happen if it does not complete executing before the next interrupt, an interlock is provided to ignore subsequent interrupts until the first is complete. This is a protection that should only be needed in debugging, but it costs little time and those bad things that happen involve corrupting the stack and making the processor run wild.

After preventing reentrance, the current application task execution interval is terminated by storing *0xFF* in the system variable *t_astrm*. Then the system state is entered, as explained later. The next step is to save the complete machine state of the interrupted task. The state is saved on the task stack, and then a special system stack is made current.

Tele measures the execution time of all tasks to a high resolution. This is described in detail later but is accomplished by a call to function *t_rtmark* immediately after storing the application machine state. All processor cycles until the next call to *t_rtmark* will be counted and accumulated under an internal task.

The initial operations take a small, fixed amount of time so that control reaches the fourth step in synchrony with the console display hardware. Function *w_cdspl* is called just as the vertical blanking interval begins so that the display can be updated rapidly without any interference appearing on the screen. The display update can last almost 4 milliseconds under some conditions. When the tick clock has a frequency higher than that of the display,

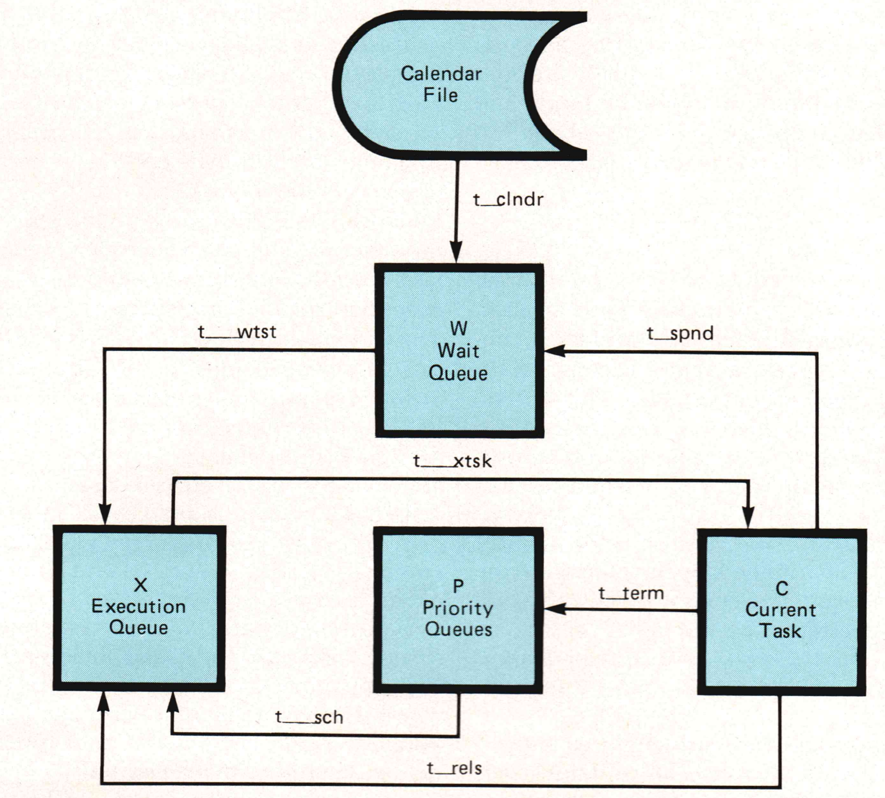


Figure 1: *Task-scheduling state machine*

(continued from page 18)

say 120 Hz, *w__cdspl* is not called on each interrupt. Because *w__cdspl* recieves control about 60 times each second, for a 120-Hz tick clock, half the interrupts bypass the display update.

The next step is to maintain the time-of-day clock. Tele provides the current time of day to application programs in an ASCII character string accurate to 0.05 second. The tick frequency is nearly a multiple of this but not exactly, so *t__tick* maintains a series of counters to skip some ticks and duplicate others. The end result is that the long-term average is very nearly 20 Hz. The time-of-day clock is intended to present the time to the operator and to time-stamp operations (it looks fast to a human being).

To simplify developing Tele under MS-DOS, I also maintain the standard BIOS time-of-day interrupt. The BIOS programs the tick clock to produce an 18.2-Hz interrupt. Tele uses some additional counters to derive this frequency from its 20-Hz clock. The standard BIOS procedure is maintained by relocating its interrupt vector before *t__tick* is installed to service the tick clock. The alternate interrupt is then invoked at the appropriate times to maintain a long-term average of 18.2 interrupts per second. Notice that at any particular instant the clocks can be off by several milliseconds—it is only the long-term (greater than 1 second) averages that are correct.

Finally *t__tick* is ready to terminate the current application task execution interval and return control to *t__krnl*. These other operations may seem complicated, but they execute very quickly and have no side effects. No problems will develop by ignoring the first several steps and assuming that control, in response to a tick interrupt, goes immediately to label *tick3*.

First, *t__rtmark* is again called to resume counting processor cycles for the interrupted task. Then *t__tick* decides whether to resume the interrupted task or to pass control back to *t__krnl*. Some tasks must not be interrupted—for instance, a system program might be rearranging the linkage between tasks. If control were to return to the task scheduler before the linkage was fully corrected, pointers could get lost and bad things happen. One way to prevent this would be to disable interrupts, but that would impinge on the device drivers. In fact, it is important to make all times that interrupts are disabled as short as possible. The time required to update the task queue linkages is much too long.

Execution States

Tele solves this problem by defining two execution states: application and system. A program enters the system

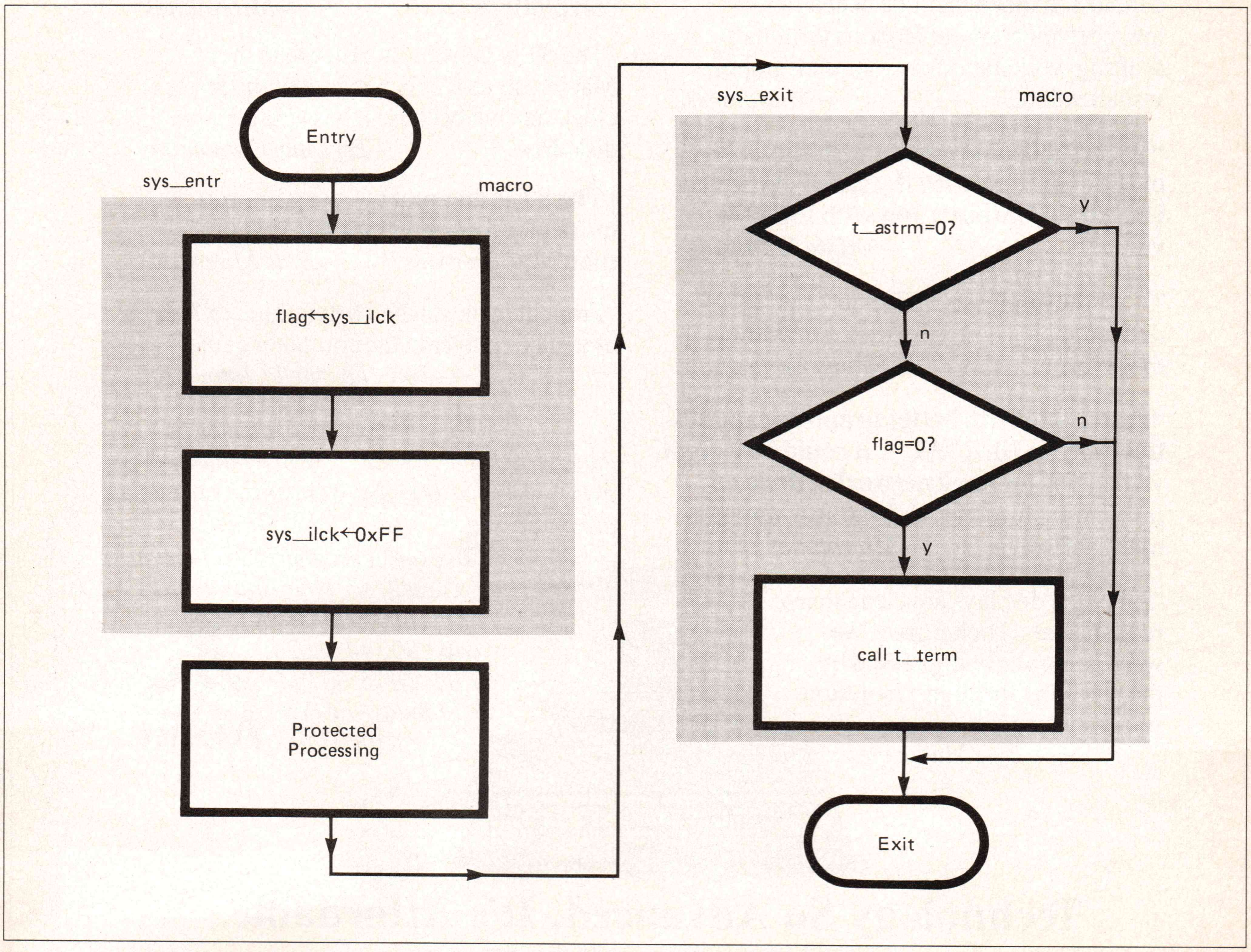


***Figure 2:** System state function*

MULTITASKING KERNEL
(continued from page 20)

state when it must be allowed to finish an operation before it is preempted. This is something like disabling interrupts, but the interrupt hardware remains fully functional and enabled. In the application state, a task may be preempted and not resumed for an indefinite period. I call the special state *system* to emphasize that it must not be entered lightly. Programs that enter it must be debugged and be sure to exit from the system state at their earliest opportunity. Figure 2, page 20, is a flowchart of a program that enters the system state.

The system state is entered with the *sys_entr* macro (see Listing Two). *0xFF* is stored in the system variable *sys_ilck* with a locked exchange instruction. The lock ensures that *0xFF* is stored and the orginal contents are loaded without any intervening bus cycles. Therefore it will work even if multiple processors are accessing a common memory. Whenever *sys_ilck* contains *0xFF*, the current program is in the system state. When it contains *0x00*, the program is in the application state. The *sys_entr* macro stores *0xFF* in *sys_ilck* and stores the original contents of *sys_ilck* in a local flag.

After the protected operation is complete, the *sys_exit* macro is used to exit from the system state. The tick service function, *t_tick*, uses the *sys_entr* macro but does not use *sys_exit*. All other functions that execute in the system state use both macros.

In the *sys_exit* macro, variable *t_astrm* is examined first. If it contains *0x00*, the current execution interval has not terminated and control returns normally. If *t_astrm* contains *0xFF*, the current application has been preempted while it was in the system state. Instead of returning control to the calling program, *sys_exit* examines the original *sys_ilck* value. If *sys_ilck* contained *0x00*, the current task is terminated by calling function *t_term* (Listing Three). But if both *t_astrm* and *sys_ilck* contain *0xFF*, the function calling the current one (where *sys_exit* is being processed) was already in the system state. Again control returns to the calling program. This scheme allows programs to be nested in the system state. Eventually control will return to the first program to enter the system state. At that time the *t_astrm* variable controls processing.

Function *t_tick* is slightly different. One difference is because *t_tick* is the only function to set *t_astrm* to *0xFF*. *T_astrm* is set to *0x00* when an application task is initially dispatched. It is set to *0xFF* when the tick interrupt occurs.

The other difference involves the stack. Every task has its own stack—at least one. The task scheduler also has its own private stack. Function *t_tick* saves the interrupted machine state on the application stack and then establishes the system stack—that is, the stack private to *t_krnl*. The system stack is then used for all subsequent processing within *t_tick*.

When control is ready to exit from *t_tick*, only the value of *sys_ilck* at the time of the interrupt need be examined (*t_astrm* is always set by this time). If it was

C spoken here...

High C™

Do you want to use a C compiler that

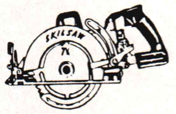
- was chosen by Ashton-Tate for dBASE III® Plus, and CAD Leader AutoDesk for AUTOCAD and AUTOSKETCH "with a twenty percent code savings over Lattice C."
- was well rated in ***Computer Language, Feb. 86:*** "Then there is High C, the most powerful compiler of all..." and ***Dr. Dobbs Journal, August 86***
- "would have saved me three weeks of porting time had I had High C instead of Microsoft's new C" *Mike LeBlanc, compiler developer, Sky Computers*
- "is the *only* C compiler for the IBM PC capable of compiling NYU's Ada/Ed compiler" *Dave Shields, research scientist, New York Univ.*
- has a complete run-time library
- has structure assignment, **enum**, **void**...
- supports nested functions as in Pascal
- supports pcc and full K&R C *plus* some latest, nifty extensions from the new ANSI-proposed C standard
- "is the highest quality C compiler for large-scale software development." *Randy Nielsen, Ansa (Paradox)*
- "saved 15% of code over five large modules of MultiMate relative to Lattice C" *David Beauchesne, Multimate International*

Pascal spoken here...

Professional Pascal™

Do you want to use a Pascal compiler that

- was chosen by Lifetree Software, Inc., for implementing Volkswriter Deluxe™
- serves as a systems and applications language at CAD/CAM giant Daisy Systems Corporation
- was well rated in ***Computer Language, May 86:*** "The clear choice for large-scale programming projects..."; and ***PC Tech Journal, July 86:*** "for team programming or for very large projects...stands absolutely alone." pg. 126 "documentation is certainly the best...a model of technical clarity." pg. 112.
- is the "howitzer" of Pascals and "could well be the most powerful Pascal compiler ever implemented on a microcomputer" ***PC Magazine, Oct. 29, 1985, p. 144***
- has 8-, 16-, and 32-bit integers; sets up to 64K bits
- has varying-strings of up to 64K characters
- has a full-fledged C macro preprocessor
- has many run-time library additions: UNIX™-like I/O, multiple heaps, interrupts,. . .
- has all the bit-pushing operators of C
- has *many* more extensions, getting you half way to Ada® for a non-Ada price



Power Tools for Power Users

Each compiler • generates **superb code**, with optimizations such as common-subexpression elimination and cross-jumping • sports no less than ***five* memory models** for the 8086 (Small, Compact, Medium, Big, and Large) • supports a unique implementation of register variables • supports the **8087/80287** in native mode, or **emulates** • supplies *three* floating-point formats • generates special instructions for the 80186/286 • generates code that runs in **80286 protected mode** • gives you **hundreds of error and warning messages**, helping you find those subtle bugs *before* you need a debugger's help • lets you **overlay *data*** as well as code (when used with PLINK86), for substantial space savings • lets you write **interrupt routines** directly in high-level language • lets you get "close to the machine" with built-in move/scan/compare operations • is supported by equivalent **resident and cross** compilers for the 80286 (UNIX V.2, Xenix, Concurrent DOS 286), 80386 (DOS, UNIX V.3), 68010/20/68881 (UNIX V & 4.2, GEM-DOS™), 32032 (UNIX 4.2), VAX (UNIX 4.2, VMS), RT PC, IBM 370,... • contains a **multi-modular cross-referencer** • produces ROM-able code for embedded applications • can talk to those **other languages** by those other vendors • gives you **64K run-time stack** space that can be shared with the heap • is endowed with an **amazing number of pragmas** (compiler controls) for customization to your application • has a compiler start-up **profile** • supports direct access to MS-DOS; library supports DOS 3.X file-sharing • generates symbolic debugger information for use with all known MS-DOS debuggers • allows **exec**-ing subprocesses • was designed for professional software developers, not hobbyists • comes with great technical support by a company that specializes in compilers • comes with extensive **typeset documentation** • and *more*... call or write for your information packet today...

Not recommended for casual use, but for applications needing **industrial-strength tools**, contact

MetaWare™
INCORPORATED
903 Pacific Avenue, Suite 201, Santa Cruz, CA 95060-4429
(408) 429-6382 (429-META), TELEX: 493-0879

Abroad:
ABC Software, The Netherlands
Microsoftware, Tokyo
Grey Matter, United Kingdom
Buchdata, Frankfurt

Professional Tools Since 1979

High C V1.3: $495 —on any MS/PC-DOS system— **Professional Pascal** V2.6: $595

OEMs: Contact us about porting our professional compilers to your systems.

TWS: Professional Compiler Developers and competitors, ask about our Translator Writing System compiler toolbox; see the review in ***Computer Language,*** *December, 1985.*

DOS Helper™: Powerful UNIX-like utilities to enhance MS-DOS, for $49.95—included free with MetaWare compilers: FIND, TAIL, MV, LS, CAT, UNIQ, FGREP, and WC.

C Validation Suite: Used by some of our competitors and many others. $2,000/Plant Site

Circle **no. 95** on reader service card.

(continued from page 22)

0x00, control simply returns into the system stack. But if *sys_ilck* contained *0xFF* at the time the tick interrupt occurred, the task was in the system state and should not be preempted. In this case, *t__tick* restores the original stack and returns into it, thereby resuming the interrupted task. That task will then finish whatever operation required system state protection and will eventually execute a *sys_exit* macro, causing control to enter function *t_term*.

Figure 1 shows the task queues and the paths along which tasks are moved from one queue to another. The paths are labeled according to the function that performs the transfer. Function *t_term* is shown moving the current task to the priority queue (from C to P). *T__tick* performs this same function.

Closely related to function *t_term* are functions *t_rels*, *t_spnd*, and *t_wait*. All these functions call another, *t__trmap*, to terminate the current application task. This is done by first storing the current machine state on the application stack. Then control enters function *t__sstk* to establish the system stack (the stack used by *t__krnl*). *T__sstk* exits by returning to the caller of *t__trmap*.

The difference between *t_term*, *t_spnd*, *t_wait*, and *t_rels* is in the value they return. Because *t__stck* has changed the stack, they return into the system stack, not the stack they were called with. Their return codes are ultimately presented to *t__krnl* and determine the path that the task will follow away from queue C in Figure 1.

Function *t_term* returns a code of 0 and causes the task to be returned to the priority queues, P. There it will compete, on the basis of priority, with other tasks to be returned to the execution queue, X. Function *t_rels* returns a code of −1, which causes the task to be returned to the bottom of the execution queue.

T_spnd returns a positive return code between 1 and 32,767. This number indicates the number of ticks for which the task will be suspended. That is, if the return

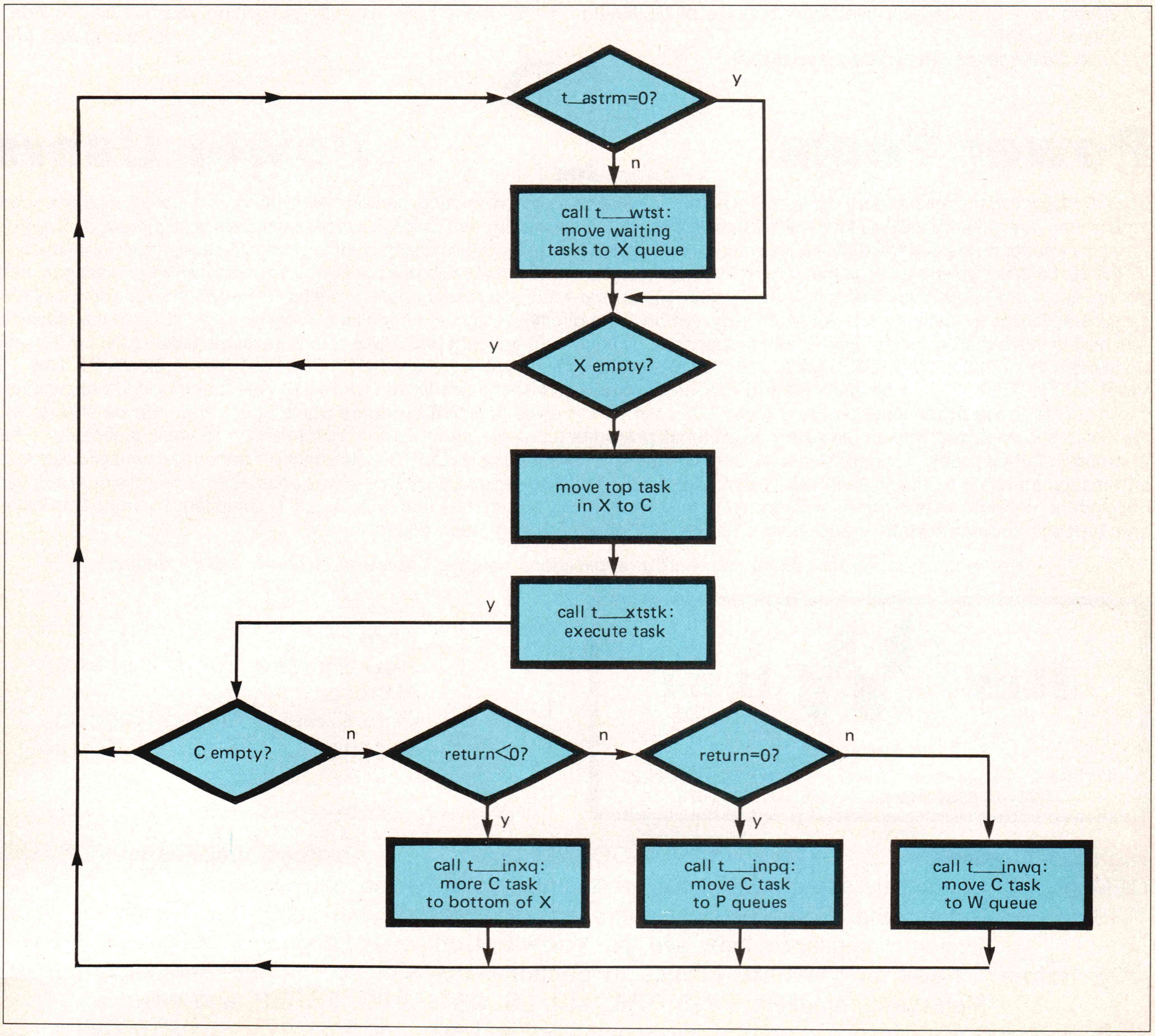


Figure 3: *Task switcher*

value is 12 and the tick frequency is 120 Hz, the task will be suspended for 1/10 second. *T_wait* is a combination; it first calls *t_spnd* and then calls *t_rels*. Therefore *t_spnd* suspends the task for a time, after which the task is placed at the top of the execution queue. *T_wait* suspends the task but restores it to the bottom of the queue.

Tele therefore has a real-time scheduling capability based on the frequency of the tick clock. The resolution is not adequate for many applications but is sufficient for times relevant to the human operator. For example, an alarm can be sounded by beeping the speaker, and the duration of the beep is measured by the tick clock using function *t_spnd*. The standard BIOS enters an idle loop that literally wastes time to measure the beep's duration, but Tele is able to continue doing other work. Similarly, many Tele drivers utilize special tasks and *t_spnd* to implement time-outs and thereby detect unresponsive equipment.

Looping

T_krnl processes the return codes from *t_term* and its associated functions. *T_krnl* is an infinite loop—once control enters, it never leaves except by calling application tasks. The application tasks eventually return to *t_krnl* and control continues to loop. Figure 3, page 24, shows the flowchart for *t_krnl*.

At the beginning of the loop *t_astrm* is examined. If it contains *0x00*, no tick interrupt has occurred. Otherwise it will contain *0xFF*, and function *t_wtst* is called. *T_wtst* will examine the wait queue (W) and move all tasks scheduled to be executed at the current system tick to the top of the execution queue. Then the execution queue is examined. If it is not empty, the task on top is made current; it is moved from X to C as shown in Figure 1. If the execution queue is empty, *t_krnl* will keep looping. Eventually a task will appear in the execution queue, and it will be executed.

Tasks are actually executed by calling function *t_xtsk*. *T_xtsk* first calls *t_rtmark* to stop accumulating processor cycles to the task scheduler. It then calls function *t_dspap*, which is the converse of function *t_trmap*. It restores an application stack and then loads the machine registers from it. Control will exit from *t_dspap* to the point following an earlier call to *t_trmap*, and the application will run until it terminates (or is terminated).

Control will then return to the point following the call to *t_dspap* in function *t_xtsk*. The return code set by *t_term* and its associated functions is then presented to *t_xtsk*, which passes it back to *t_krnl* when *t_xtsk* returns. Before returning, *t_xtsk* again calls *t_rtmark*. This time the call causes subsequent processor cycles to be accumulated to the task scheduler.

T_xtsk is able to determine the actual execution time of the application task. This can be much less than a full system tick interval if the application was interrupted many times. If the true execution time is less than a certain threshold, *t_xtsk* immediately executes the task again. This is done only if the task was preempted. If the task terminated itself (by calling *t_spnd*, *t_rels*, or *t_wait*), no check of execution time is made.

If the task had a fair chance at the processor, or terminated itself, control returns from *t_xtsk* to *t_krnl*. Now the return code established by *t_term* and the others is examined. If the return is less than 0, function *t_inxq* is called to move the task to the bottom of the execution queue. If the return is 0, function *t_inpq* is called to move the task to the priority queues. Otherwise function *t_inwq* is called to insert the task into the wait queue. That completes the task scheduler loop, which is often called a monitor or supervisor cycle in other operating systems.

Multitasking System Techniques

Tele uses its central task-scheduling algorithm to support other features. Figure 1 indicates that function *t_sch* moves tasks from the priority queue to the bottom of the execution queue. This function is not called by any other task. It is placed in the execution queue when the system is initialized. When it executes, it enters the system state and processes the priority queues, taking tasks from P to the bottom of X. It places itself at the bottom of the execution queue last. Therefore, when all the tasks it has scheduled have been run, it will run again itself in order to schedule more.

It is easy to extend this concept to several versions of *t_sch*, each operating on a different set of priority queues. Some IBM mainframe operating systems obtain a similar effect with programs called initiators.

A similar function is *t_clndr*, which lives in the wait queue. It provides for long-term calendar functions by periodically executing and then rescheduling itself in the wait queue. As shown in Figure 1, the wait queue accounts for a short time—about 30,000 ticks. At a 120-Hz tick clock rate, this is about 4.5 minutes. *T_clndr* executes once every minute. It examines an extended calendar file on disk, and as the time to execute programs nears, it creates tasks in the wait queue.

High-Resolution Clock

Tele measures the execution time of every application task. It also measures the amount of time spent processing the system tick interrupt and scheduling the next task. This is done with a resolution of 15 microseconds.

A standard IBM PC uses one channel of an 8257 DMA (direct memory access) controller to refresh dynamic memory. Dynamic memory chips are based on leaky capacitors, which means they must be read periodically in order to maintain their contents (when read, the chip automatically recharges its capacitors). Most dynamic memory chips can be properly refreshed by reading 128 consecutive addresses every 2 milliseconds. Most PCs do this by programming one channel of the same 8253 counter to produce a signal with a 15-microsecond period. This signal then requests the DMA controller to read the memory. Special circuitry causes all memory chips to be read at once (the data read is ignored). The DMA controller automatically increments its address after each request.

The processor can read the current address in the DMA controller at any time. Because the refresh circuitry runs continuously from the time the computer is powered up, the current address is a convenient high-resolution clock.

In standard PCs, the processor clock and 8253 clock are both derived from the same crystal. They therefore maintain a constant phase difference. When the tick clock is

(continued from page 25)

started, the current reading of the high-resolution clock is saved. This allows programs to relate the high-resolution clock to the current time of day and make absolute time measurements to within 15 microseconds.

Tele itself only uses the high-resolution clock to count processor cycles. A refresh cycle is 72 processor cycles on a 4.77-MHz processor—it is more on faster ones. Therefore the number of processor cycles measured can vary from run to run. The count tends to become more accurate as longer runs are measured.

Function *t_rtmark* (Listing Four) is provided to read the high-resolution clock. It keeps track of the previous reading so that each time it is called it accumulates the interval just terminated toward some task. Tele mostly measures execution time to document its actions. This data is useful in tuning the system and diagnosing some application problems. The only regular use made of it is in rerunning tasks that get short execution intervals because of heavy interrupt service.

Installation

Tele is designed to be a collection of programmers' tools. The listings with this article are suitable for inclusion with small-memory-model C programs. To make them work, you must assemble and compile them into a library. Then you can reference these functions from your own programs.

I developed the code using the Lattice C compiler, Version 2, and Microsoft MASM assembler, Version 1. I later upgraded to Versions 3 and 4 and had to make minor changes. If you use Lattice C, Version 3, you should include the switches *−w −cc* in the LC1 command line. This specifies word alignment of variables and nested comments. If you use another compiler, you must ensure these conditions and may have to make other changes as well. The assembly-language programs contain macros (*pseg*, *endps*, *dseg*, and *endds*) defined with the Lattice

Books on Operating System Design

The principles and practice of operating system design cannot be communicated in one magazine article. The two books reviewed here should be on the desk of any operating system builder. Prentice-Hall has been doing great things with computer-science textbooks during the last few years, publishing a series of texts that are both well written and practically oriented. All too often, computer-science texts are neither. They're incomprehensible unless you're a mathematician, and from reading the book, you'd have no idea that the subjects covered had any practical application. In fact, the only thing that's difficult with many computer-science topics is understanding the books. This review discusses two welcome exceptions to the incomprehensibility rule, both on the subject of operating system design.

Biggerstaff, Ted J. *Systems Software Tools*. Englewood Cliffs, N.J.: Prentice-Hall, 1986.

Systems Software Tools is about multitasking operating systems. Over the course of the book, Ted Biggerstaff develops a small multitasking kernel that supports up to four concurrently executing processes, each running in its own window. The system runs on an IBM PC, but the book is not really targeted at IBM programmers. For the most part, DOS is used as an I/O system rather than as an operating system. Though a certain amount of space is devoted by necessity to IBM-specific topics, it's easy to port both the concepts and the code to a different environment.

The book breaks up its subject into the same layers that are found in the operating system itself, organized from the machine outward. That is, the earliest chapters talk about how to interface to the actual hardware, and the subject develops gradually to the user interface, presented in the last chapter.

Biggerstaff starts out with a quick summary of the C programming language. It's not a tutorial, but enough of the language is covered that you'll be able to follow the code in the remainder of the book (provided that you know a language such as Pascal pretty well). There's also a discussion of DOS interfacing conventions and how to use the DOS I/O system. These chapters are lucid and cover all the basics of DOS interfacing. They're pretty useful in their own right. Because the main thrust of the book isn't DOS interfacing, it's good that the author has concentrated all the DOS-specific stuff in one place.

Systems Software Tools starts really moving in Chapter 4, which discusses interrupt processing and communications hardware. In this and the next chapter, Biggerstaff develops a low-level, interrupt-driven console I/O system, discussing such topics as I/O queues and writing interrupt service routines. The I/O system is exercised with a terminal emulator program that works directly with the hardware, bypassing DOS entirely. Chapter 6 is a discussion of concurrent operating systems in general, explaining how multitasking works and covering most of the essential topics, such as scheduling strategies and interprocess communications. The basic data structures, such as task control blocks, are developed in Chapter 7, and Chapter 8 discusses process management—how to get two programs to run at the same time and how to transfer control from one to the other. Finally, the last chapter ties it all together and presents a viable user interface built around a simple windowing system.

Biggerstaff's operating system, though it's pretty nifty, does have a few flaws. It doesn't do low-level disk I/O but uses DOS system calls when necessary. This approach is pragmatic because a primitive disk I/O system is readily available for most machines, but it's not much help if you want to learn how to put a disk I/O system together from scratch. Because all the code is written in

compiler. These macros define the proper segment structure for linking with the C modules.

Installation is accomplished by calling function *t_init* (Listing One). When control returns, your program will be executing as a single task, and others may be executed at the same time. You can create additional tasks by calling function *t_crt*. They can be destroyed by calling *t_del*. See Listing One for the calling procedures.

It's important to call function *t_term* before returning to MS-DOS. If you don't, an interrupt vector will be left pointing into the transient program area. It is likely that a hard crash will result the next time a program is loaded unless you call function *t_term* to restore the original vector.

The last part of Listing One contains a *main* function. This tests the task scheduler by creating two subtasks. Each subtask continually increments a counter, and the program displays the current values on the display. Though *main( )* and *count( )* are not part of the task scheduler, they do serve as an example of its use.

Conclusion

Tele's task scheduler actively participates in the execution of every task, but it is almost always transparent. Most tasks can assume they are running under a single-tasking system. They can take advantage of services associated with multitasking, such as intertask communication, without significantly altering their structure.

An expanded version of this code is available from me for $100. It's available as a programmer's tool and includes full source code and precompiled libraries for all 8086 memory models, more detailed documentation than is possible here, and diagnostic functions useful for debugging modifications.

Further information is available from Berry Computer, P.O. Box 966, Jackson, CA 95642; (209) 223-0993.

DDJ

(Listings begin on page 50.)

Vote for your favorite feature/article.
Circle Reader Service **No. 2.**

C, the system response times are slower than need be, too. I'm not sure what the context-swap time is, but it's probably too slow for many real-time applications. On the other hand, the C is much more readable than the equivalent assembler would be, and once you know the theory, translating the C to assembler is not too difficult. Finally, the system presented is not compatible with anything. This isn't necessarily a problem—for example, the code would ROM quite nicely if you want to build a little stand-alone system that runs on a single-board computer.

On the other hand, *Systems Software Tools* is a very good introduction to operating system design in general. The system presented is pretty useful, in spite of its flaws, and you've got the entire source code if you want to make changes. I recommend this book to anyone who's already a reasonably proficient programmer and wants a good introduction to very-low-level systems programming. A knowledge of C and a little assembler is useful but not essential. The book is readable and well organized, and all the subjects covered have immediate application. The code is well written and nicely illustrates the theoretical concepts presented in the text. Moreover, a complete program is presented that you could type into your machine and run (you can also get the code on IBM-compatible disk). The code is most applicable to IBM PC-based machines, but the operating system itself is portable to just about any environment.

Comer, Douglas. *Operating System Design, The XINU Approach.* Englewood Cliffs, N.J.: Prentice-Hall, 1984.

Douglas Comer's *Operating System Design, The XINU Approach* is a good complement to Biggerstaff's book. It's more advanced and covers most of the topics that Biggerstaff omits. On the other hand, Comer presents the operating system only—he offers absolutely no utilities, not even a shell or user interface.

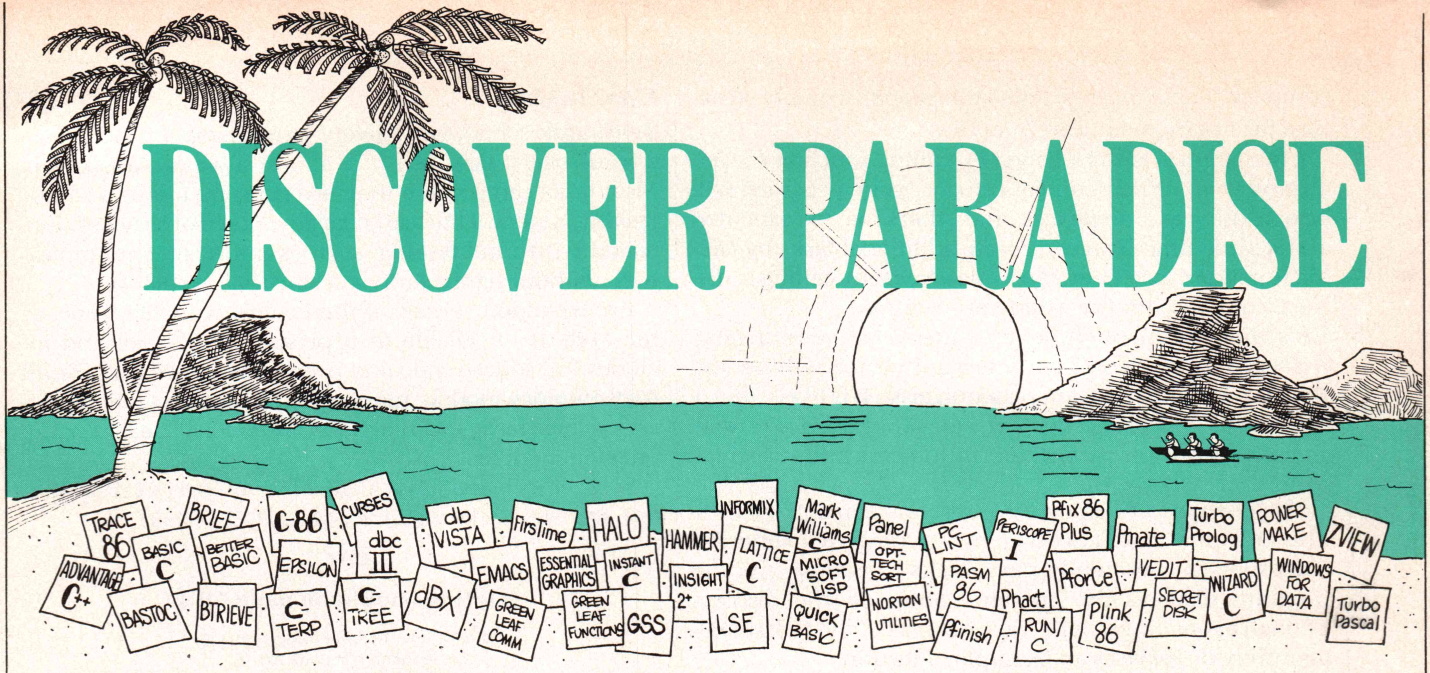
XINU, a stand-alone operating system that includes both a disk I/O system and file server, is presented in its entirety. As with Biggerstaff's book, you could type in the code and have a complete operating system. XINU stands for "Xinu Is Not Unix," and the name is apt. XINU is a scaled-down Unix. All the essential parts of the kernel are there, and they are functionally very similar to their Unix equivalents. There's a Unix-like device-driver mechanism, and the disk is organized much as Unix organizes its own disk. The code presented is not Unix source code, however; it's Comer's implementation of that code. XINU is not a toy—it's a complete operating system that should be useful in virtually any application you might cook up (with the possible exception of real-time control systems).

XINU was originally written for an LSI 11/02, but it contains virtually no machine-specific code and so is quite portable. It was developed on a larger machine and downloaded to the target machine. There's almost no assembly language in XINU; the overwhelming majority of the code is in C. Its disk I/O system interfaces to a Xebec S-1410 5¼-inch Winchester controller. The Xebec presents a pretty standard hardware interface, and the techniques presented should port to most other controllers with little difficulty.

Comer's book is too involved to dissect chapter by chapter. Like Biggerstaff, he's organized the chapters in terms of functional layers, but he covers many more layers. Comer covers the basic stuff in a somewhat cursory manner that might be confusing if you've never seen any of the material before. He also doesn't present as much theory as I'd like, limiting himself to the implementation of a specific operating system rather than to discussing operating systems in general. On the other hand, XINU is a powerful, complete operating system and it's all there for you to examine.

Both of these books are good—Biggerstaff's is more introductory and Comer's more complete—but taken together they provide a good introduction to operating system design. I recommend them highly.—*Allen Holub*

DISCOVER PARADISE



Programmer's Paradise Gives You Superb Selection, Personal Service and Unbeatable Prices!

Welcome to Paradise. The MS/PC-DOS software source that caters to your individual programming needs.

Discover the Many Advantages of Paradise...

- **Lowest price guaranteed.**
- Only the **latest versions** of all products.
- **Large inventory** of popular products ready for immediate shipment.
- Product evaluation **literature** available.
- **Special orders**—If you don't see your product listed, call.
- **30 day money-back guarantee**—We will refund your money for products returned in fully resalable condition within 30 days of purchase. Please ask for details when placing an order.

How much extra would you expect to pay for all this special service? At Paradise, it's yours for the asking. **Call our toll free number—1-800-445-7899.** Our software pros are ready and waiting to help you.

We'll Match Any Nationally Advertised Price.

	LIST	OURS
C++		
ADVANTAGE C++	$ 495	CALL
C COMPILERS		
C-86 COMPILER	395	279
C-86 PLUS COMPILER	497	CALL
DESMET C COMPILER		
W/SOURCE LEVEL DEBUGGER	159	145
LATTICE C COMPILER	500	294
W/LIBRARY SOURCE	900	545
LET'S C	75	58
W/CSD SOURCE		
LEVEL DEBUGGER	150	109
MICROSOFT C 4.0	450	288
MARK WILLIAMS C	495	289
SUPERSOFT-C	395	339
WIZARD C	450	359
C INTERPRETERS		
C-TERP	300	235
INSTANT C	495	379
INTRODUCING C	125	104
RUN/C	150	89
RUN/C PROFESSIONAL	250	169
C UTILITY LIBRARIES		
ASYNC MANAGER	175	135
BASIC_C	175	129
C ESSENTIALS	100	85
C-FOOD SMORGASBORD	150	98
W/SOURCE	300	188
C TOOLS PLUS	175	135

	LIST	OURS
C UTILITY LIBRARY	185	135
DATA WINDOWS	225	159
ENTELEKON COMBO PACKAGE	200	169
C FUNCTION LIBRARY	130	109
C WINDOWS	130	109
SUPERFONTS FOR C	50	43
ESSENTIAL COMM LIBRARY	185	139
W/ BREAKOUT DEBUGGER	250	199
GREENLEAF FUNCTIONS	185	134
GREENLEAF COMM	185	134
THE HAMMER	195	149
MULTI C	149	135
PFORCE	395	239
WINDOWS, SCREEN DISPLAY, GRAPHICS		
CURSES	125	94
W/SOURCE CODE	250	184
ESSENTIAL GRAPHICS	250	205
FLASH UP WINDOWS	75	68
HALO	300	209
META WINDOWS	185	115
META WINDOWS PLUS	235	189
METAFONTS	80	59
METAFONTS PLUS	235	189
MICROSOFT WINDOWS	99	68
MICROSOFT WINDOW DEV. SYS.	500	329
ON-LINE HELP	149	109
PANEL	295	219
WINDOWS FOR DATA	295	250
SCREENPLAY (LATTICE C)	150	135
TOPVIEW TOOLBASKET	250	189

	LIST	OURS
VIEW MANAGER	275	199
VITAMIN C	150	135
VC SCREEN	99	84
ZVIEW	245	189
ARTIFICIAL INTELLIGENCE		
LISP		
BYSO LISP	395	CALL
EXPER LISP	495	445
IQC LISP	300	269
GOLDEN COMMON LISP	495	CALL
GC LISP 286 DEVELOPER	1195	CALL
MULISP	250	175
WALTZ LISP	169	149
PROLOG		
ARITY PROLOG (STANDARD)	95	59
ARITY PROLOG INTERPRETER	350	329
ARITY PROLOG INTERPRETER		
AND COMPILER	795	739
ADDIT. ARITY PRODUCTS	CALL	CALL
CHALCEDONY PROLOG/i	100	CALL
PLA MICROPROLOG (INTRO)	99	89
W/APES 1.1	149	135
PLA PROFESSIONAL MICROPROLOG	395	345
W/APES 2.0	650	489
TURBO PROLOG	100	75
OTHER AI		
APES 2	395	339
EXPERT CHOICE	495	449
ESP ADVISOR	895	839

	LIST	OURS
EXSYS	395	319
INSIGHT 2+	485	379
EXPERTEACH II	475	379
LOGIC LINE SERIES	CALL	CALL
METHODS	79	69
SMALLTALK V	99	89

BASIC

	LIST	OURS
BETTER BASIC	199	139
8087 MATH MODULE	99	79
BTRIEVE INTERFACE	99	79
C INTERFACE	99	79
RUNTIME MODULE	250	219
BETTER TOOLS	95	89
FINALLY	99	89
MICROSOFT QUICKBASIC	99	75
PROFESSIONAL BASIC	99	75
8087 MATH SUPPORT	50	45
PANEL-BASIC	145	115
RM/BASIC	600	479
TRUE BASIC	150	105

OTHER PRODUCTS AVAILABLE TO THE BASIC PROGRAMMER INCLUDE MULTIHALO, BTRIEVE, GSS GRAPHICS, SCREEN SCULPTOR, STRUBAS, 87 BASIC.

COBOL COMPILERS/UTILITIES

	LIST	OURS
MICROSOFT COBOL	700	445
MICROSOFT COBOL TOOLS	350	205
MICROSOFT SORT	195	139
OPT-TECH SORT	149	115
REALIA COBOL	995	785
SCREENPLAY	175	155
RM/COBOL	950	639
RM/COBOL 8X	1250	895

Featured Product of the Month

MICROSOFT C COMPILER v. 4.0—Latest version of the "... best MS-DOS C development environment value today."—Dr. Dobbs. Provides fast, optimized code. Version 4.0 includes Microsoft's new Codeview source-level windowing debugger. List $450 Ours **$285**

ASSEMBLERS/LINKERS

	LIST	OURS
ADVANTAGE LINK	495	CALL
MACRO 86	150	98
PASM 86	195	129
PLINK 86 PLUS	495	329

MAKE/PROFILE/LINT

	LIST	OURS
LMK	195	149
POLYMAKE	99	79
OTHER POLYTRON PRODUCTS	CALL	CALL
PMAKER	125	89
PFINISH	395	239
THE PROFILER	125	95
PC LINT	139	105
PRE C	295	159

Terms and Policies

- **We honor MC, Visa, American Express** No surcharge on credit card or C.O.D. Prepayment by check. New York State residents add applicable sales tax. Shipping and handling $3.00 per item, sent UPS ground. Rush service available, prevailing rates
- Programmer's Paradise will match any current nationally advertised price for the products listed in this ad.
- **Mention this ad when ordering**—some items are specially priced
- Prices and Policies subject to change without notice.
- Corporate and Dealer inquiries welcome.

DEBUGGERS

	LIST	OURS
ADVANCED TRACE-86	175	138
BREAKOUT	125	99
CODESMITH 86	145	108
C SPRITE	175	138
CI PROBE	75	59
CSD SOURCE DEBUGGER	75	59
PERISCOPE I	295	245
PERISCOPE II	145	109
PERISCOPE II-X	115	84
PFIX 86 PLUS	395	239

EDITORS

	LIST	OURS
BRIEF	195	CALL
CVUE	75	59
W/SOURCE	250	195
EMACS	295	265
EPSILON	195	159
FIRSTIME (C)	295	229
KEDIT	125	105
LSE	125	95
PMATE	195	119
PC/Vi	149	129
SPF/PC	195	149
VEDIT	150	109
VEDIT PLUS	225	139

FILE MANAGEMENT

	LIST	OURS
BTRIEVE	245	195
XTRIEVE	245	195
RTRIEVE	145	119
BTRIEVE/N	595	464
XTRIEVE/N	595	479
RTRIEVE/N	345	279
C TREE	395	329
DBC III	250	189
W/SOURCE	500	378
DB VISTA	195	155
W/SOURCE	495	425
INFORMIX	995	795
PHACT	295	265
QUERY	195	175
REPORT	165	149
PHACT W/SOURCE	995	845
OPT-TECH SORT (FOR BTRIEVE)	149	115

FORTRAN COMPILERS

	LIST	OURS
F77L LAHEY FORTRAN	477	CALL
MICROSOFT FORTRAN	350	208
PROSPERO FORTRAN	390	319
RM/FORTRAN	595	389

New Products

C-86 PLUS—All new proposed ANSI standard C compiler using artificial intelligence technology to produce really fast executable code. List $497 Ours **CALL**

DATA WINDOWS—Greenleaf's latest offering includes integrated windows, transaction data entry, pop-up, pull-down, and Lotus style menu systems. And more! Data Windows is fast, writing directly to video memory. List $225 ($450 w/source) Ours **$159** ($315 w/source)

ADVANTAGE LINK—The first overlay linker to take advantage of extended memory. It supports memory caching object file merging complex overlay structures and automatic overlay reloading. List $495 Ours **CALL**

TIMESLICER—Discovery!!! Linkable C library which provides Multi-tasking and Real-Time functionality at the application level. Call for information and/or price.

FORTRAN UTILITIES

	LIST	OURS
ACS TIMES SERIES	495	419
87 SFL	250	225
FOR-WINDS	90	78
FORLIB-PLUS	70	54
GRAFMATICS OR PLOTMATICS	135	119
GRAFMATICS AND PLOTMATICS	240	219
FORTRAN SCIENTIFIC SUBROUTINES	295	249
POLYFORTRAN TOOLS I	179	143
STRINGS AND THINGS	70	54

ALSO AVAILABLE TO THE FORTRAN PROGRAMMER: PANEL, MULTIHALO, BTRIEVE, ESSENTIAL GRAPHICS, FLASH UP WINDOWS, GSS GRAPHICS, OPT-TECH SORT.

PASCAL COMPILERS

	LIST	OURS
MICROSOFT PASCAL	300	189
PROSPERO PASCAL	390	315
TURBO PASCAL	100	69
OTHER BORLAND PRODUCTS	CALL	CALL

December Holiday Bundles

When you buy **MICROSOFT C** or **LATTICE C** you also get any one of the following: **GREENLEAF FUNCTIONS, GREENLEAF COMM LIBRARY, C UTILITY LIBRARY, ESSENTIAL COMM LIBRARY** FOR ONLY **$399**

TOOLS FOR MS PASCAL

	LIST	OURS
ASYNC MANAGER	175	135
ESSENTIAL GRAPHICS	250	205
EXEC	95	75
FIRSTIME	245	199
FLASH UP WINDOWS	75	68
GSS GRAPHICS DEV. TOOLKIT	495	395
HALO	300	209
PANEL	295	219
PASCAL TOOLS I	125	99
PASCAL TOOLS II	100	79
SCREEN SCULPTOR	125	95
VIEW MANAGER	275	199

TOOLS FOR TURBO PASCAL

	LIST	OURS
ALICE	95	68
FIRSTIME	75	59
FLASH UP WINDOWS	75	68
HALO	300	209
SCREENPLAY	100	89
SCREEN SCULPTOR	125	94
TURBO PASCAL ASYNC MGR	100	84
TURBO PROFESSIONAL	70	49
TURBO POWER TOOLS PLUS	100	83

TRANSLATORS/BRIDGES

	LIST	OURS
BASTOC (MBASIC)	495	399
C TO DBASE	150	135
DBC III	250	189
W/SOURCE	500	378
D BX	350	329
FORTRIX	6000	CALL

ADDITIONAL PRODUCTS

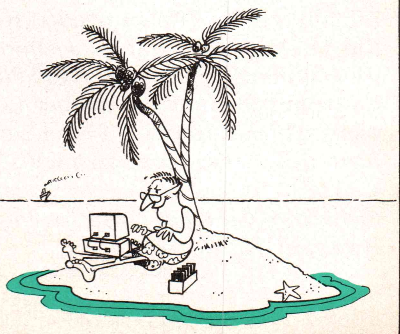
	LIST	OURS
DAN BRICKLIN'S DEMO PROGRAM	75	59
FASTBACK	175	149
INTERACTIVE EASYFLOW	150	129
PDISK	195	129
TEXT MANAGEMENT UTILITIES	120	94

1-800-445-7899

In NY: 1-800-642-6471

Programmer's Paradise
487 E. Main Street, Mt. Kisco, NY 10549
914-332-4548

Programmer's Paradise



Circle **no. 334** on reader service card.

In Search of a Sine

by Richard A. Campbell

> ***I was unable to find any useful algorithms for computing mathematical functions.***

I was recently involved in a project for which I needed to compute sines and cosines of angles on the NS320xx microprocessor with a floating-point coprocessor. I looked through several years' worth of the more erudite microcomputer magazines (including, I must admit, *DDJ*), some mathematics and computer-science textbooks, and the source code of some programs that included sine computation routines. Despite my searching, though, I was unable to find any useful algorithms for computing mathematical functions in general and sines in particular. I was looking for an algorithm explained in simple terms, along with a reasonably well-commented program for carrying it out.

I did find several potential answers, but for various reasons none of them were usable. Some of them were too vague or mathematically complicated for me to understand well enough to program them. Another was an uncommented program for the 8080 CPU in which the sine computation algorithm was obscured by the floating-point arithmetic routines. One was an uncommented program written in STOIC (a Forth-like, stack-oriented language) that was impossible to figure out well enough to recode it. Another involved exponentiation of e, which was computationally impossible for me.

The Taylor Series

I finally got the hints I needed from the Mathematical Tables section of the *Handbook of Chemistry and Physics* from the Chemical Rubber Company. There I found a Taylor series formula for the sine of an angle. The series can be expressed as:

$$s = a-(a^3/3!) + (a^5/5!) - (a^7/7!)\ldots$$

where a is the angle (in radians), s is the sine, and ! is the operator for factorials (the factorial of an integer n is the product of n with all integers smaller than itself and greater than 1). Because the factorials of relatively small numbers can be calculated easily (for example, $4! = 2\times3\times4 = 24$), this seemed like a simple enough algorithm. I coded it up in BASIC and tried it out by computing the sines of several angles, starting at 0° or radians and comparing the results with the values in the Mathematical Tables, which are printed to five decimal places.

It became obvious that it was computationally faster to precompute coefficients that included the value of 1 divided by the factorial and the sign of the term. Thus, the series above became:

$$\begin{aligned} s = a &-(0.166666 \times a^3) \\ &+(0.00833333 \times a^5) \\ &-(0.0001984127 \times a^7)\ldots \end{aligned}$$

Working with this test program revealed that the number of terms needed to approach the accuracy of the five-place tables was dependent on the size of the angle. For angles up to 90°, five terms were needed (through a^9 divided by 9!). This seemed computationally reasonable. But to get up to 120°, seven terms were needed (through a^{13} divided by 13!). At this point I realized this approach was not feasible because an algorithm that can't compute the sines of large angles is of little use.

Quadrants of the Circle

The way out of this dilemma is to consider the angles in quadrants by thinking of the sines starting at 0° and progressing around the circle. As the angle increases from 0° to 90°, the sines go up from 0 to +1. From there to 180°, the sines return to 0, pass down to −1 at 270°, and then return to 0 at 360°. Thus the sine of an angle between 90° and 180° is equal to the sine of 180° minus the angle. Table 1, page 31, shows a summary of all this. If the quadrant is determined, you need only be able to compute sines from 0° to 90° efficiently.

Doing It in BASIC

When I considered the quadrant phenomenon, the uncommented programs I had discovered began to make more sense. One was particularly interesting because it seemed to involve only four coefficients. It's in the STOIC floating-point-routine file (CP/M Users' Group) and is credited to J. Sachs, 1977. Frankly, I'm not sure I've interpreted Sachs' approach as intended, but the following does work.

First, scale the incoming angle to quadrants such that an angle of 90° (1.570795 radians) is made to equal 1. If you're working in radians, this means dividing by 1.570795; if you're using degrees, divide by 90. Then, if

Richard A. Campbell, 198 Washington Hwy., Snyder, NY 14226

the scaled angle is greater than 4 (360°), subtract 4 from it and repeat the comparison (and subtraction) until it's less than 4. Then compare with 3 (270°) and 1 (90°) and adjust as appropriate based on Table 1.

Compute the sine approximation as:

$$s = (C1 \times a) + (C2 \times a^3) + (C3 \times a^5) + (C4 \times a^7)$$

where the coefficients are:

C1 = 1.570795
C2 = −0.645921
C3 = 0.07948765
C4 = −0.004362469

Remember that the standard Taylor series approximation has the input angle expressed in radians; these are scaled to proportions of 90°. So the coefficients are the same as those of the standard series divided into 1.570795 and raised to the appropriate power, rather than 1. For example, C2 is −1.570795 to the third power divided by 3! (approximately).

The actual computation, in BASIC, is given in Table 2, page 32. Sines calculated in this way will usually agree with the table values; some are off by 1 on the fifth place. This seems like a tolerable error. Runs of 1,000 sines of angles from 0° to 100° in 0.1° steps indicate that each sine takes about 28 milliseconds to compute on a 6-MHz NS16032 CPU.

The above algorithm can be used for cosines simply by doing the scaling and then adding 1 (adding 90°, actually), then computing the sine. Compute the tangent by computing sine and cosine and then dividing sine by cosine (checking for zero cosine, of course.)

Doing It in Assembler

The NS320xx assembly-language routine (Table 3, page 32) is about 210 bytes long, including the coefficient constants and the temporary storage space. Runs calculating 10,000 functions of angles from 0° to 100° in 0.1° steps indicate that sines and cosines take 0.9 milliseconds to compute and tangents take 1.3 milliseconds on a 6-MHz 16032 CPU. The results, rounded to five decimal places, generally agree with the table values; occasionally errors of 1 in the fifth place occur with sines and cosines. Tangent errors are slightly greater; sometimes errors of 2 in the fifth place happen.

The binary values of the constants in the program were determined by setting BASIC variables equal to the desired numbers and then examining the symbol table in memory with a debugging program.

Reading the Listing

The motion of arguments in NS320xx instructions is from left to right. The CPU and the coprocessor each have eight registers, numbered *R0* through *R7*. The instruction *MOVBF 1,R1*, for example, takes a byte-sized integer 1, converts it to floating-point format, and leaves it in coprocessor register (CPR) *1*. *DIVF R2,R3* divides the operand in CPR *3* by the operand in CPR *2* and leaves the quotient in CPR *3*.

The instruction *MOV.D 0(R0)[R6:D],SNTAB* may need some explanation. The meaning may not be obvious because it involves an ad-

sine of 0° to 90°	= sine of angle
sine of 90° to 180°	= sine of 180°-angle
sine of 180° to 270°	= −sine of angle-180°
sine of 270° to 360°	= −sine of 360°-angle

Table 1: *Range reduction for* SIN(x)

COMPUTING SINES
(continued from page 31)

dressing mode not available on most microprocessors: It performs a 32-bit move from the address in *R0* plus zero indexed by (plus) the contents of *R6* multiplied by 4 (the length of the moved data) into *SNTAB*. Thus *R6* contains the entry number in *SNTAB*—not a byte displacement but rather a double word displacement—and *R0* contains the base address of the table.

Another possibly confusing instruction is *MULF R5,SNTAB*. In this case, you might think it would be more efficient to move the coefficient into a register rather than to use memory. I had previously discovered, however, that the use of CPU registers (address and/or index) and a coprocessor register in the same instruction often leads to chaos. This is not pointed out well in National Semiconductor's documentation.

The rest of the program is fairly straightforward and should be directly applicable to any CPU that is comparable to the NS320xx in power.

DDJ

Vote for your favorite feature/article.
Circle Reader Service **No. 3.**

```
APOW = AN : REM AN = SCALED ANGLE
SINE = AN × C1 : REM C'S AS ABOVE
APOW = APOW × AN : REM FIGURE AN ^3
APOW = APOW × AN
SINE = SINE + APOW × C2 : REM ADD TERM
APOW = APOW × AN : REM FIGURE AN ^5
APOW = APOW × AN
SINE = SINE + APOW × C3 : REM ADD TERM
APOW = APOW × AN : REM FIGURE AN ^7
APOW = APOW × AN
SINE = SINE + APOW × C4 : REM ADD TERM
```

Table 2: *Polynomial approximation in BASIC*

```
; Listing One: Computing sine, cosine, and tangent
; NS-320XX assembler
; Sine, cosine, & tangent routines
RadScl   MOVF    PIO2,R2       ;Divide by pi/2 = 1.57079635
         BR.S    DoScal
DegScl   MOVBF   90,R2         ;Scale for degrees
; Scale for sin cos, tan
DoScal   DIVF    R2,R3         ;Now 90 degrees = 1.00
DQuadr   MOVBF   4,R4          ;4 = 360 degrees
         MOVBF   1,R1          ;1 = 90 degrees
         MOVBF   2,R2          ;2 = 180
         MOVBF   3,R5          ;3 = 270
SnScl1   CMPF    R3,R4         ;Is angle > 360 degrees?
         BLT.S   SnScl3        ;go if less than 360
         SUBF    R4,R3         ;else subtract 360
         BR.S    SnScl1        ;do it again
SnScl3   CMPF    R3,R5         ;>270?
         BLE.S   SnScl5        ;no, go
         SUBF    R4,R3         ;make minus: ang=ang-360
SnScl5   CMPF    R3,R1         ;>90?
         BLE.S   ScnCl7
         SUBF    R3,R2
         MOVF    R2,R3         ;ang=180-ang
ScnCl7   RET                   ;scaled value in R3
ACos     MOVBF   90,R2         ;angle, degrees in R3
         DIVF    R2,R3         ;divide by 90
         MOVBF   1,R2
         ADDF    R2,R3         ;make angle plus 90
         BSR     DQuadr        ;figure quadrant
         BR.S    DoSin         ;go do sine, return from it
ASin     BSR     DegScl        ;Angle in degrees
                               ;Fall through to. . .
; Compute Sine(x)
; Sine = 0 + 1.570795*X -0.645921*X^3
;          + 0.07946765*X^5 - 0.004362469*X^7
DoSin    ADDR    SNTAB,R0      ;get address of table
         MOVQ.D  4,R6          ;init R6
; Enter with table addrs in R0, # terms in R6
         MOVF    R3,R1         ;save angle in R1
         MOVF    R1,R5         ;and in R5
         MOVBF   0,R3          ;sine starts at zero
SinLp    MOV.D   0(R0)[R6:D],SNTAB ;get multiplier
         MULF    R5,SNTAB      ;mult angle
         ADDF    SNTAB,R3      ;add to sine
         MULF    R1,R5         ;angle ^ n+2
         MULF    R1,R5
         ACB.BS  -1,R6,SinLp   ;do again till done
         RET
; Compute Tan(x)
; Tan(x) = Sin(x)/Cos(x)
ATan     MOVBF   90,R2         ;scale for degrees
         DIVF    R2,R3
         MOVF    R3,SNTEMP     ;save scaled angle
         BSR     DQuadr        ;figure quadrant
         BSR     DoSin         ;compute sin
         MOVF    R3,TOS        ;save sine on stack
         MOVBF   1,R3
         ADDFF   SNTEMP,R3     ;add for cosine
         BSR     DQuadr        ;figure quadrant
         BSR     DoSin         ;compute cosine
         MOVF    R3,R2         ;move cosine to R2
         MOVF    TOS,R3        ;recover sine
; R0 now = 0
         CMPF    R0,R2         ;have zero?
         BNE.S   ATanDv        ;no, divide
         MOVF    FNSM,R3       ;else use big number
         RET
ATanDv   DIVF    R2,R3         ;Tan = Sin/Cos
         RET
FNSM     BYTE    0,0C0h,0DAh,45h     ;7000
PIO2     BYTE    0DBh,0Fh,0C9h,3Fh   ;pi/2 = 1.57079635
SNTEMP   BLK.D   1             ;temporary storage
SNTAB    BLK.D   1             ;temp
; these are in reverse order of use
         BYTE    0Bh,0F3h,8Eh,0BBh   ;-0.004362469
         BYTE    6Ch,0CAh,0A2h,03Dh  ;0.07948765
         BYTE    14h,5Bh,25h,0BFh    ;-0.645921
         BYTE    0D0h,0Fh,0C9h,3Fh   ;1.570795
```

Table 3: *Computing* TAN(x) *from* SIN(x)/COS(x)

"How to protect your software by letting people copy it."

By Dick Erett, President of Software Security

Inventor and entrepreneur, Dick Erett, explains his company's view on the protection of intellectual property.

"A crucial point that even sophisticated software development companies and the trade press seem to be missing or ignoring is this:

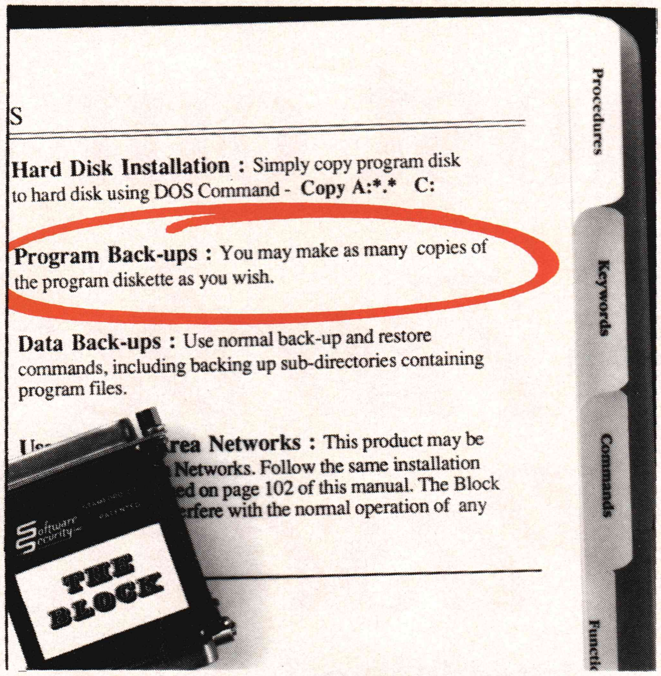
Software protection must be understood to be a distinctively different concept from that commonly referred to as copy protection.

Fundamentally, software protection involves devising a method that prevents unauthorized use of a program, without restricting a legitimate user from making any number of additional copies or preventing program operation via hard disk or LANs.

Logic dictates that magnetic media can no more protect itself from misuse than a padlock can lock itself.

Software protection must reside outside the actual storage media. The technique can then be made as tamper proof as deemed necessary. If one is clever enough, patent law can be brought to bear on the method.

Software protection is at a crossroads and the choices are clear. You can give product away to a segment of the market, or take a stand against the theft of your intellectual property.



Soon all software installation procedures will be as straightforward as this. The only difference will be whether you include the option to steal your product or not.

"...giving your software away is fine..."

We strongly believe that giving your software away is fine, if you make the decision to do so. However, if the public's sense of ethics is determining company policy, then you are no longer in control.

We have patented a device that protects your software while allowing unlimited archival copies and uninhibited use of hard disks and LANs. The name of this product is The BLOCK™.

The BLOCK is the only patented method we know of to protect your investment. It answers all the complaints of reasonable people concerning software protection.

In reality, the only people who could object are those who would like the option of stealing your company's product.

"...eliminating the rationale for copy-busting..."

Since The BLOCK allows a user to make unlimited archival copies the rationale for copy-busting programs is eliminated.

The BLOCK is fully protected by federal patent law rather than the less effective copyright statutes. The law clearly prohibits the production of work-alike devices to replace The BLOCK.

The BLOCK attaches to any communications port of virtually any microcomputer. It comes with a unique customer product number programmed into the circuit.

The BLOCK is transparent to any device attached to the port. Once it is in place users are essentially unaware of its presence. The BLOCK may be daisy-chained to provide security for more than one software package.

Each software developer devises their own procedure for accessing The BLOCK to confirm a legitimate user. If it is not present, then the program can take appropriate action.

"...possibilities... limited only by your imagination..."

The elegance of The BLOCK lies in its simplicity. Once you understand the principle of The BLOCK, hundreds of possibilities will manifest themselves, limited only by your imagination.

Your efforts, investments and intellectual property belong to you, and you have an obligation to protect them. Let us help you safeguard what's rightfully yours. Call today for our brochure, or a demo unit."

Software Security inc.

870 High Ridge Road Stamford, Connecticut 06905

203 329 8870

Circle **no. 170** on reader service card.

PAINLESS WINDOWS.

Windows. Data Entry. Menus. Finally, a C programmers' tool that makes them as easy to use as *printf()*. With Greenleaf DataWindows™, you move in quantum leaps!

Snazzy Window Treatments

DataWindows represents an important breakthrough in C programming tools. It sets you free so you can create exciting programs quickly and easily, saving both time and money! Developed to work with the IBM PC, XT, AT, compatibles, and MSDOS or PCDOS, DataWindows is a carefully tooled system of C functions which will jazz up your programs with unprecedented efficiency.

Greenleaf DataWindows is integrated windows, transaction data entry, pop-up, pull-down, and Lotus style menu systems with:

- **Screen Management.** You don't have to remember what's on the display or the sequence in which you put it there. DataWindows does the grunt work. There are no restrictions.
- **Transaction Data Entry.** Data entry windows can have any number of fields with sophisticated options for reading many data types. Calls are made to help, validation, and other functions. Full featured text editing, protected and mandatory fields, dBASE type picture strings, context sensitive help, validation of fields and transactions, redefinable keys, password entry, attribute control, keyboard idle and much more.
- **Device Independence.** It detects the type of display adapter your computer is using and adjusts to it automatically for CGA, EGA, or monochrome. Logical video attributes are easy to use for color or monochrome.
- **Compatibility.** Runs with Microsoft Windows and IBM TopView.
- **The Greenleaf Tradition of Quality.** Reliable products. Professional documentation that gets you up and running quickly and keeps you there. Reference card. Newsletter and Bulletin board.

IBM, Microsoft & dBase, are registered trademarks of International Business Machines, Microsoft Corporation & Ashton-Tate respectively. PCDOS, IBM PC, XT, AT, & TopView are trademarks of IBM; MSDOS and Microsoft Windows are trademarks of Microsoft Corporation.

Stop Window Shopping

Order Today. Or call toll free for a free demo of the windows library that makes all the others obsolete.

Order any of these high performance tools by calling your dealer or 1-800-523-9830 today. Specify compiler when ordering. Add $8 for UPS second day air, or $5 for ground. Texas residents add sales tax. MasterCard, VISA, P.O., check, COD. In stock, shipped next day.

Greenleaf DataWindows	$225
DataWindows Source Module	$225
The Greenleaf Comm Library v2.0	$185
The Greenleaf Functions v3.0	$185
Digiboard Comm/4-II	$325
Digiboard Comm/8-II	$535

GREENLEAF Software©

1411 LeMay Drive, Suite 101
Carrollton, TX 75007

Call Toll Free
1-800-523-9830
In Texas and Alaska, call
214-446-8641

Circle no. **97** on reader service card.

Window Dressings

- **Simple or Complex Windows.** Up to 254 powerful overlaid windows simultaneously, all with just one kind of window to remember! Yet any window can be from one character to 32K!
- **Easy Window Operations.** DataWindows lets you move, zoom, frame, title, change colors, titles, frames, size, location, and make windows visible or invisible at will! Functions set cursor, attributes, and write data to any window or "current window". Word wrap, auto scroll, keyboard functions.
- **Write to Any Window Any Time.** Windows may be visible, overlaid, or invisible, and you can write to them anyway. What you write will be seen when the windows become visible.
- **DataWindows is fast!** It writes directly to video memory (in some modes).
- **Easy to save!** Any window, complete with attributes, can be saved on disk quickly and efficiently.
- **Source code available. No royalties.**

Also from Greenleaf:

The Greenleaf Functions v3.0
The most complete, mature C language function library for the IBM PC,XT,AT and close compatibles. Includes over 225 functions — DOS, disk, video, color text and graphics, string, time/date, keyboard, disk status and Ctrl-Break functions plus many more.

The Greenleaf Comm Library
Our 2.0 version is the hottest communications facility of its kind. Over 120 functions — ring buffered, interrupt driven asynchronous communications for up to 16 ports simultaneously with XMODEM, XON/XOFF, many many sophisticated features.

We support all popular C compilers for MSDOS/PCDOS: Microsoft, Lattice, Computer Innovations, Aztec, DeSmet, and others.

Echelon's Z-System

by Morris Simon

Z-System is an 8-bit operating system for Z80 systems.

Richard Conn's public-domain ZCPR3 is an enhanced substitute for CP/M 2.2's console command processor (CCP). Echelon, a software firm headed by public-domain pioneer Frank Gaudè, has combined ZCPR3 with Dennis Wright's ZRDOS, a Z80-optimized BDOS replacement, to form the joint nucleus of an 8-bit operating system supported by a cluster of customized tools, utilities, libraries, and hard-copy manuals. The term *Z-System* refers specifically to the operating system itself (ZCPR3 combined with ZRDOS) and generically to Echelon's entire line of ZCPR3 utilities and software tools.

In this article, I'll evaluate the ZCPR3-ZRDOS nucleus of Z-System from the viewpoint of an advanced programmer, stressing its program design and structure.

Versions of Z-System

Echelon sells Z-System in two basic forms: Joseph Wright's auto-install version, Z-Com, which uses a host CP/M 2.2 system to overlay the CCP and BDOS with ZCPR3 and ZRDOS and then expands the host CBIOS to make room for memory-resident utilities and buffers; and a manual-install or "hacker" version consisting of the source codes for all system components and utilities except the optional ZRDOS. The SIG/M public-domain version omits the proprietary ZRDOS but comes with Conn's SYSLIB3 and derivative Z3LIB and VLIB libraries of Z80 routines used in the assembly of Z-System components.

Z-Com ($119) is the better choice for turnkey systems, inexperienced users, or anyone who wants to install a complete Z-System in a few minutes. Advanced programmers who will be doing major customizations to Z-System segments have both the development tools and the assembly-language skills needed to install the more expensive but flexible manual version ($182.50 with ZRDOS, if purchased piecemeal from Echelon).

The major difference between the two versions is that Z-Com requires the presence of CP/M 2.2 to boot it as a transient command file. In contrast, the manual installation procedure completely replaces the CP/M CCP and BDOS on the system tracks with ZCPR3 and ZRDOS (if present) to produce a bootable Z-System disk. In addition to Z-Com's use of CP/M to boot itself, the two versions differ in that Z-Com's system components are preselected and contained in a set of object-code files. With Z-Com, major changes and additions can be tricky and may require some disassembly and patching. Z-Nodes, a remote bulletin-board network of international Z-System users, distributes useful public-domain patch files to make this job a little easier.

The manual-install version comes with the source code for all major system components (except the optional proprietary ZRDOS) so that users can modify any feature of ZCPR3 separately to suit their needs or hardware requirements. Assembly of the altered code requires a good relocatable macro assembler, preferably one that can assemble Zilog mnemonics, and standard system alteration and debugging utilities such as MOVCPM, SYSGEN, and DDT or their equivalents. Modification and reassembly of the individual utilities may also require access to the appropriate Z-System source-code libraries—SYSLIB3, Z3LIB, and VLIB.

Components

Each version uses its own installation procedures and design principles to achieve the same functional result. In either case, the user's CBIOS is modified and extended by several optional memory-resident system segments—customized packages of Z80 code and buffer areas. A complete Z-System includes

- ZCPR3, the console command processor
- either BDOS or ZRDOS
- a modified CBIOS
- a revised page zero with jump addresses to the new CBIOS and BDOS or ZRDOS and new buffers (in the manual version)

The modified CBIOS contains

- the original BIOS codes
- an initializing routine that sets Z-System equates and addresses
- the Resident Command Package (RCP) (a cluster of memory-resident utilities)
- the Input/Output Package (IOP) (space reserved for customized I/O drivers)
- the Flow Command Package (FCP) (equates used by system utilities that permit conditional logic and flow states)

Morris Simon, 118 Brookhaven, Tuscaloosa, AL 35405

• the Environment Descriptor (Z3ENV) (system equates used by utilities)
• buffers, stacks, and equates (reserved spaces for shell operations, messages shared by several utilities, named directories, RAM registers, and assorted byte or word locations such as the "wheel" security byte)

Figures 1 and 2, below, illustrate memory-map differences for the two versions using sample addresses from a TeleVideo 803 64K RAM. Segment sizes and locations will vary with other systems.

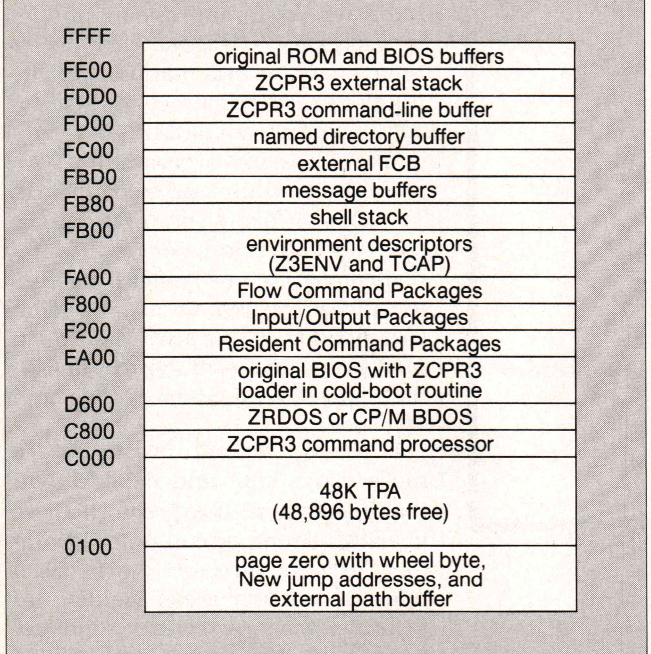
The auto-install version makes Z-System transportable to a wide variety of memory structures because it involves less alteration of the user's CBIOS than the manual version does. With Z-Com, the entire Z-System package is loaded in front of the original CBIOS, extending it and overwriting both the CP/M BDOS and CCP. A new jump table at the first RAM page of the relocated CBIOS intercepts system calls and redirects them to Z-System segments such as the Input/Output Package or to the original CBIOS jump locations. The system segments and buffers are loaded between the new and old CBIOS jump tables. ZRDOS and the ZCPR3 command processor reside below this extended CBIOS, at somewhat lower addresses than CP/M's CCP and BDOS did in the host system.

When I first started using Z-System, I was disturbed by this loss of vital RAM space, as any serious 8-bit user should be. I soon realized, however, that most CP/M 2.2 programs are written so tightly that they seldom require a TPA of more than 40K or so. This is a tribute to 8080 and Z80 assembly-language programmers who can never afford the 16-bit luxury of bulky and redundant code—there's just no room for it. Even when CP/M programs overwrite the command processor, they often are just using the BDOS (or ZRDOS) address stored at location 0006H in page zero as a convenient way to set up a program stack that will work on any CP/M system. The TPA between the end of a program and the end of its stack is often an empty space of more than 20K, whereas the Z-System extensions require only around 6K.

The manual-install version shown in Figure 1 saves a little more RAM space (512 bytes) by moving the CBIOS downward, placing the external path buffer and wheel byte in page zero and relocating the original BIOS jump table. If ZRDOS is installed, it resides directly beneath the original BIOS jump table, just as the BDOS does in CP/M systems. All ZCPR3 external segments and buffers reside above the original CBIOS, except for an initializing routine poked into the used cold-boot loader space.

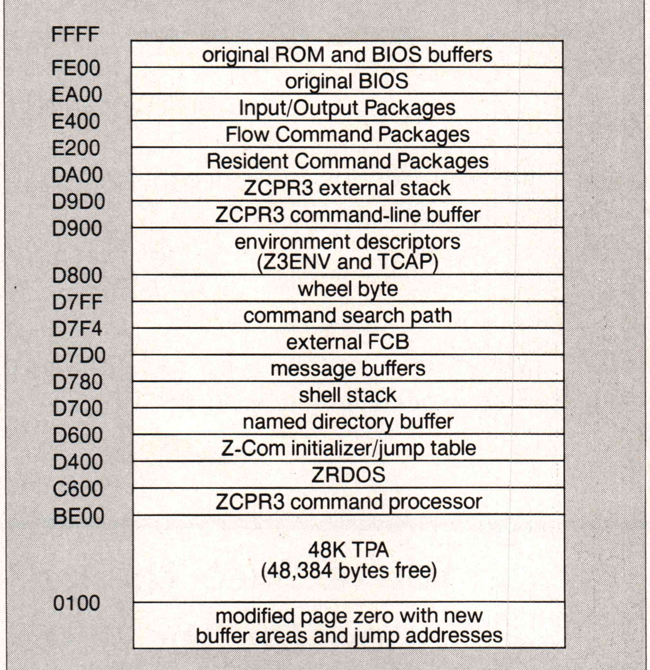
ZCPR3 and Its Extensions

The heart of either Z-System version is Conn's ZCPR3 command processor with its internal enhancements and CBIOS extensions. ZCPR3 users have many optional and enhanced internal commands, such as *SAVE*, *GO*, *JUMP*, *TYPE*, and others, that are unavailable in CP/M 2.2. Multiple and



Hexadecimal addresses are system-dependent and optional. This example is based upon a composite of Conn's model in *ZCPR3: The Manual* and my own Z-System segments within a TeleVideo CBIOS.

Figure 1: *A complete manual-install version of Z-System*



Hexadecimal addresses are system-dependent and optional. This example is based upon one of my own system images using a 64K Tele Video 803 CBIOS.

Figure 2: *A complete auto-install (Z-Com) version of Z-System*

(continued from page 37)

extended command processing, instantaneous user area or directory selection, command path assignment, and wheel access control are some of the internal ZCPR3 options that are often found only on larger systems. By skillful assembly-language programming, Conn has squeezed these and other features into the original 2K CCP space in order to maintain compatibility with earlier CP/M programs for which the CCP size is critical. Although these enhancements in the body of the command processor are definitely valuable, ZCPR3's unique advantages lie in the external Z-System segments.

ZCPR3's modular components reside in high memory for fast access, increasing the power and flexibility of the command processor beyond that of any other 8-bit operating system and for some operations of larger ones as well. These fast resident utilities actually permit users of tiny 8-bit, 64K systems to enjoy the speed and efficiency of a RAM disk for many routine tasks.

The modular design of the ZCPR3 CBIOS extensions permits users to modify any resident utilities, named directory structures, flow control states, input/output drivers, command paths, and most other Z-System features whenever they choose to change or add to them.

With the manual-install version, you can perform massive alterations upon any of these features simply by editing and reassembling one or more well-documented source files into a variety of relocatable object-code segments. With the Z-Com version, you may have to disassemble some object code in order to tailor the system segments to suit your needs. When either object-code package is ready, you can use a fast loader utility (LDR.COM) to reinstall different system segments instantly to fit changing jobs.

You might, for example, wish to use separate resident command packages (RCPs) for word processing and assembly-language programming, with different resident utilities in each one to reduce disk access time. I have several manuscript-editing aids in a word-processing RCP and a fine memory editor (MU3) in another RCP I use for programming. An alias batch command reconfigures my CBIOS by loading one of these specialized command packages automatically each time I enter its respective directory area. Because resident command packages always reside in RAM, I can run any of these utilities at high speed without modifying any TPA code.

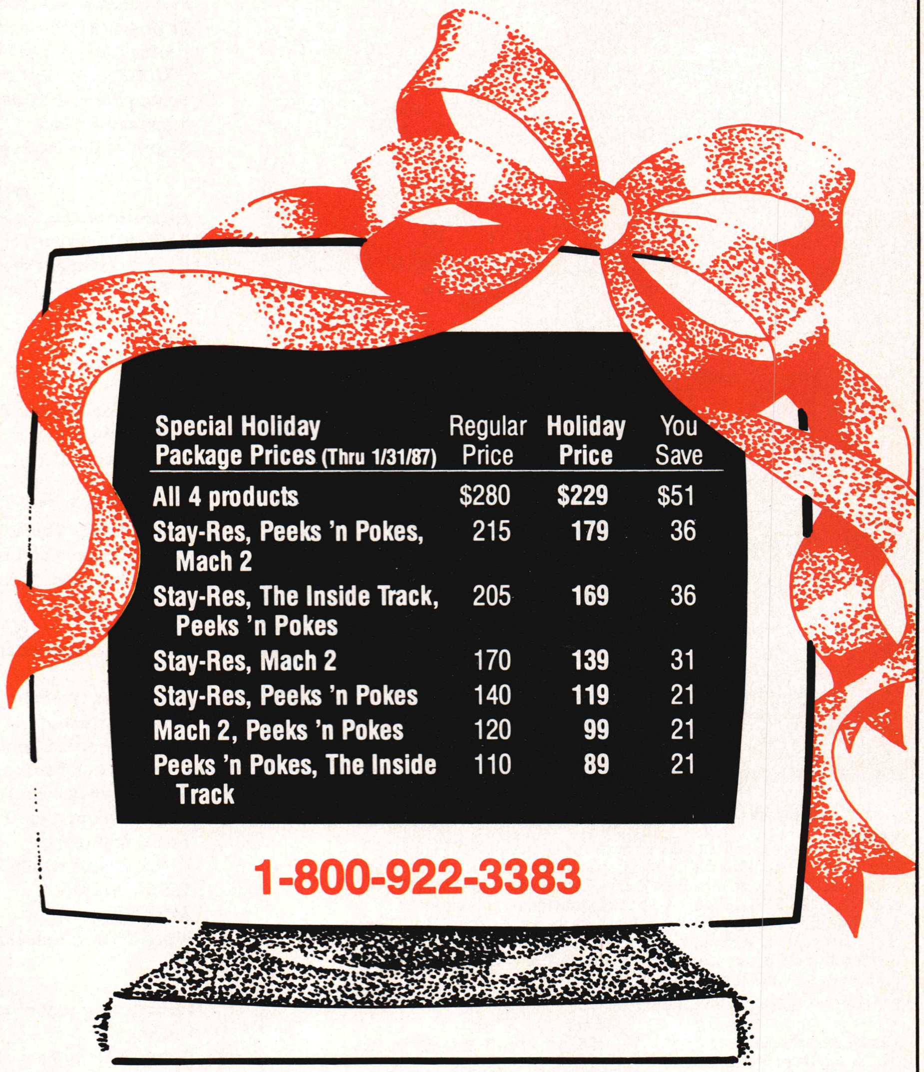
Z-System's named directories are more convenient and flexible than the Unix-style MS-DOS directories I use in an IBM-equipped computer laboratory. ZCPR3's directories are based upon CP/M's user areas, yielding up to 32 possible directories or subdirectories with optional passwords on any one "level." You need to expand the default named directory buffer if you plan to use more than 14 directory names at one time.

All ZCPR3 directories exist on the same level to the operating system, but you can create a Unix-style hierarchical effect by overlaying previous levels automatically with the change

30 Day Money Back Guarantee
(Call for Details)

Holiday Gift Ideas—

For Yourself or Your Friends



Special Holiday Package Prices (Thru 1/31/87)	Regular Price	Holiday Price	You Save
All 4 products	$280	$229	$51
Stay-Res, Peeks 'n Pokes, Mach 2	215	179	36
Stay-Res, The Inside Track, Peeks 'n Pokes	205	169	36
Stay-Res, Mach 2	170	139	31
Stay-Res, Peeks 'n Pokes	140	119	21
Mach 2, Peeks 'n Pokes	120	99	21
Peeks 'n Pokes, The Inside Track	110	89	21

1-800-922-3383

STAY-RES—Make Compiled BASIC* Programs Memory-Resident.

Invoke your program with a "hot key" or a POKE from another program. Protects you from calling DOS functions at unsafe times. Automatic screen save/restore in text or MONO/CGA graphics modes (including EGA, Hercules). More than one program can be co-resident. Enables you to perform multi-tasking with BASIC. Compatible with any well-behaved program and most not-so-well behaved ones. Demo available. $95

MACH 2—Ultra-fast, Extensive Assembler Subroutine Library for BASIC*

Our assembler subroutines give your programs a crisp, professional look and feel. Display data instantly when compiled. Window manager. Store, sort, search string data using all available DOS memory—no more 64K limit. Controlled input routine allows full use of standard editing keys & ignores Ctrl-C and Ctrl-Break. Change screen attributes without re-displaying text. Read/write files FAST as DOS. PRINT USING for numbers up to 6 times faster than BASIC. Number rounding. DOS/BIOS function calls & interrupts. Many more functions. Source code available. Demo available. No assembler or knowledge of assembler programming is required. $75

PEEKS 'N POKES—Get More out of the IBM PC Family*

Read & change system configuration. Read/write CMOS chip on AT class machines. Unprotect GW-BASIC, BASIC, BASICA programs saved with "P". Read & change keyboard status. Plug characters into keyboard buffer. Read printer, COM and video status. Swap monitors, COM ports and printers. Generate more and better sounds than BASIC. Find more PEEKS & POKES. Tutorials on binary & HEX numbers, DEF SEG, Port, INP, OUT, logical operators. Boot your system three different ways from BASIC. Many more functions. Most source code included. $45

THE INSIDE TRACK—Programmer's Utilities & Starter Assembler Routine Library for BASIC*.

Control keyboard shift status from DOS or your program—force CAPS lock, NUM lock, etc. Keep diskette motor running for better throughput. Copy protect a diskette. Boot your system three different ways from DOS. Duplicate one monitor's contents on the other monitor. Load & execute large EXE files faster than DOS, from DOS or compiled BASIC. Limit memory used by compiled BASIC programs to avoid reloading Command.com. Send ANSI control characters to screen in any compiled BASIC. Memory map and BIOS call reference chart, including EGA calls. All source code included. Many more functions. $65

NOT COPY PROTECTED. None of our software is copy protected. All subroutines can be included in your compiled programs with no royalties. All software requires an IBM PC, XT, AT, Tandy 1000, 1200, 3000 or compatible with DOS 2 or later. Free technical support included with all products.

*Each package includes a manual containing complete instructions and sample programs. All packages are compatible with QuickBASIC 1 & 2, BASCOM 1 & 2, MS BASIC Compiler 5.36. (Stay-Res with QB 2 requires DOS 3 or later.) Stay-Res programs use a minimum of about 90K Memory. All except Stay-Res compatible with BASIC Interpreter, BASICA, GW-BASIC, MS BASIC.

DEMONSTRATION DISK: Send us a check for $5 (no credit cards, please) and we'll send you a demo of Mach 2 and Stay-Res. The $5 will be applied to your purchase of any of our software that is purchased directly from us.

ORDER NOW. VISA/MC/COD orders call 1-800-922-3383 (COD USA only—add $3). In Georgia (404) 973-9272. Add shipping and handling: $3 per order USA, $5 for 1 or 2 packages to Canada (add $2 each addt'l package). Elsewhere $18 for first package, $7 each additional. GA residents please add sales tax.

MicroHelp, Inc.
2220 Carlyle Drive • Marietta, Georgia 30062

Circle **no. 215** on reader service card.

PC Network: the world's #1 supplier of

*members pay wholesale plus 8% and get 14–30 day software rentals

***PC NETWORK—Members pay just 8% above the wholesale price, plus shipping.** All prices reflect a 3% cash discount. Please add $1.00 for each software package ordered. Remember, the minimum shipping charge is $2.50. Money orders, personal and company checks please allow 10 working days to clear. All prices subject to change without notice.

†**RENT BEFORE YOU BUY**—Members are eligible to join the NETWORK'S Business and Game software Rental Libraries and evaluate products for a full 14 (Regular) or 30 (VIP) days to see if it meets your needs. And The NETWORK'S rental charges are far less than other software rental services—Just 20% OF THE MEMBERS WHOLESALE PRICE. **Rental titles** available in IBM/Apple/Mac and CP/M Formats.

CALL TOLL FREE

(800) 621-SAVE

(Orders—Membership and Advice!)
In Illinois Call (312) 280-0002
Your Membership Validation Number DD22

You can validate your membership number and, if you wish, place your first money-saving order over the phone by using your VISA, MASTERCARD or AMERICAN EXPRESS.
Our knowledgeable sales consultants are on duty Mon.-Fri. 8:00 AM to 7:00 PM Sat. 9:00 AM to 5:00 PM CST.

PERSONAL COMPUTER NETWORK
320 West Ohio Street
Chicago, Illinois 60610
Customer Service and Order Status (312) 280-1567
Starting Nov. 1, 8:00 AM to 8:00 PM, Mon.-Fri.

PC NETWORK MEMBERSHIP APPLICATION

YES! Please enroll me as a member in the PC NETWORK™ and send my catalog featuring thousands of computer products, all at just 8% above DEALER WHOLESALE PRICES. I will also receive "THE PRINTOUT", a special quarterly update on merchandise at prices BELOW even those in my wholesale catalog and all the other exclusive, money-saving services available to Members. I am under no obligation to buy anything. My complete satisfaction is guaranteed.

Please (✓) all boxes that apply:

	1 Year	2 Year
•Basic Membership With 14 Days Rental	**$8** ☐	**$15** ☐
Business Software Rental Library	**$25** ☐ add'l. per year	
Games Software Rental Library	**$10** ☐ add'l. per year	
•Special V.I.P Membership With 30 Days Rental	**$15** ☐	**$25** ☐
BOTH Business and Game Software Rental Libraries	**$30** ☐ add'l. per year	

☐Bill My Credit Card: ☐VISA ☐MasterCard ☐American Express
Account Number: ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Exp. Date ____________
mon. yr.

☐Check or Money Order Enclosed for $ ____________
Name ____________
Address ____________ Apt. No. ______
City ____________ State ____________ Zip ______
Telephone ( ) ____________
My Computer(s) is: ☐IBM PC ☐IBM XT ☐IBM AT ☐Apple II ☐Macintosh Other ______

Signature ____________
(Signature required to validate membership)

OVER 30,000 PRODUCTS AT

Arity Corporation

Expert System Development Package	$235.00*
File Interchange Toolkit	37.00*
Prolog Compiler and Interpreter	650.00*
Prolog Interpreter	275.00*
SQL Development Package	235.00*
Screen Design Toolkit	37.00*
Standard Prolog	70.00*
Arity Combo Package	950.00*

Blaise Computing Inc.

Asynch Manager for C	$115.00*
Asynch manager for Pascal	115.00*
C Tools/Tools 2 Combo Pack	115.00*
C Tools	85.00*
C Tools	65.00*
C Tools Plus	115.00*
Exec	65.00*
Pascal Tools/Tools 2 Combo	115.00*
Pascal Tools	85.00*
Pascal Tools 2	65.00*
Runoff	35.00*
Turbo Asynch Plus	65.00*
Turbo Power Tools Plus	65.00*
View Manager w/Source for C	185.00*
View Manager w/Source for Pascal	185.00*

Borland International

Reflex	$54.00*
Reflex and Workshop Bundle	99.95*
Reflex Workshop	35.25*
Turbo Database Toolbox	34.00*
Turbo Editor Toolbox	35.00*
Turbo Gameworks Toolbox	35.00*
Turbo Graphix Toolbox	32.97*
Turbo Lightning	48.50*
Turbo Pascal w/8087 and BCD Support	50.00*
Turbo Tutor	19.50*
Word Wizard	37.25*
Word Wizard and Turbo Lightning Bundle	79.50*

Lattice

C-Compiler 3.1	$250.00*
C-Sprite Debugger	115.00*
C-Food Smorgasbord	82.00*
Sidetalk	79.00*
DBC-III Library	137.00*
RPG-II Compiler	410.00*

LogiTech

LogiMouse	$75.00*
LogiMouse +	89.00*
LogiMouse + w/PC Paint	117.00*
LogiMouse + w/Generic CAD	139.00*
LogiMouse + w/Reflex	145.00*
LogiMouse + w/Generic CAD and PC Paint	158.00*
Modula–2/86 Compiler	55.00*
Modula–2/86 Compiler w/8087	85.00*
Modula–2/86 Compiler w/512K	142.00*
Modula–2 Runtime Debugger	41.00*
Modula–2 Source Package	79.00*
Modula–2 Translator	35.00*
Modula–2 Utilities Package	35.00*
Modula–2 Window	35.00*

```
;   ["Z3-Dot-Com" is the auto-install version of ZCPR3
;   alone while "Z-Com" includes ZRDOS as well]

;   Z3-Dot-Com and Z-Com
;   Copyright 1985 by Joseph W. Wright

Z3ENV           SET         0DC00H
;   No other equates should be changed!
;   [This is the address of the environment
;   descriptor. Addresses and segment sizes are more
;   critical in Z-Com than in the manual-install
;   version. Note that all other addresses are
;   offset from the environment descriptor.]

;   General equates
false       equ         0
true        equ         not false
base        equ         0

i8080       equ         false           ;   [for 8080 systems]

expath      equ         z3env-0ch       ;   [command path]
expaths     equ         5               ;   [5 2-byte elements]

z3whl       equ         z3env-1         ;   [wheel byte address]

rcp         equ         z3env+200h      ;   [address of Resident
rcps        equ         16              ;   Command Package and
                                        ;   16-block size]

iop         equ         z3env+0c00h     ;   [address of Input/
iops        equ         12              ;   Output Package and
                                        ;   12-block size]

   . . .                                ;   [code continues to
                                        ;   define all segments]
```

Table 1: *Excerpt from Z-COM version of Z3BASE.LIB. Comments in square brackets are mine.*

ECHELON'S Z-SYSTEM

(continued from page 38)

directory (*CD*) and alias commands. This causes a new named directory system to be loaded into the CBIOS and reorganizes all the files contained in the new user areas you tell the system to "see." The result can be a virtual hierarchy of dynamic subdirectories, or a flat passworded subdirectory layer, or whatever you wish it to be. As with the other modular extensions of the command processor, the package of directory names can be changed by simple editing, and a special make directory (MKDIR) utility is provided just for that purpose.

The buffers and locations used by the various components of Z-System must be initialized at the time ZCPR3 and ZRDOS (if present) are loaded. In the manual version, a configuration routine resides in the CBIOS cold-boot loader area from which it assigns all ZCPR3 internal and external options as well as external buffers and other CBIOS locations. Changing parameters, addresses, or buffer sizes is a simple matter of editing two header files, Z3HDR.LIB and Z3BASE.LIB, and making whatever secondary changes are required elsewhere, especially when you have changed the size of a segment buffer or two.

In the manual-install version, Z3HDR.LIB contains users' options for most ZCPR3 features and commands and Z3BASE.LIB defines the locations of external segments. In Z-Com's version of Z3BASE.LIB (Table 1, above), these addresses are all defined as offsets from a single environment descriptor location determined by the installation process. You can still poke or patch changes in the object code of Z-Com components if you know what you're doing.

When assembled, the Z3BASE.LIB header file becomes the first half of ZCPR3's environment descriptor, Z3ENV, which gives Z-System a major transportability advantage over other 8-bit operating systems. The descriptor is merely a duplex header file containing default or user-defined information such as the addresses of all Z-System segments, CRT and printer values, CPU speed, cursor and highlighting controls, and many other system-specific details.

All Z-System utilities use the de-

scriptor's data to customize their operation and to communicate with the system segments. In order to install a new utility inside your operating system, you merely place a pointer to Z3ENV in the utility's source code and refer all system-dependent routines to it. Table 2, below, illustrates this use of the environment descriptor in the source code for a fast file-find utility.

Calls to the Terminal Capabilities (TCAP) half of the environment descriptor permit the immediate transportability of any software that uses terminal-specific features such as video graphics, windowing, and pull-down menus. Users can customize their programs instantly to take advantage of special terminal features such as graphics characters and cursor positioning codes. Neither software nor hardware differences will interfere with transportability of Z-System programs as long as the Z3ENV pointer is properly installed. A special utility, Z3INS.COM, can install or reinstall the Z3ENV address quickly in any object-code files written for Z-System use.

ZCPR3's flow control and multiple command features are essential components of shells and interactive menus. Both permit users to construct long conditional arguments that are useful in the automation of complex tasks. ZEX (an extended command processor), VFILER (a file-sweep

```
        . . .
; This is the address of the customized Z3ENV
Z3ENV   SET   0FE00H
; These are external macros found in Conn's SYSLIB
; and Z3LIB libraries
        EXT   Z3INIT,GETNDR,GETWHL,CODEND
        EXT   COUT,CRLF,PRINT,PADC
; This conditional assembly statement tests the
; code for the address of an external Z3ENV.
        IF  Z3ENV NE 0                 ; External Z3ENV found.
        JP  START                      ; Go to first instruction.
        DB  'Z3ENV'                    ; Label indicating that
                                       ; this is a ZCPR3 utility.
        DB  1                          ; Code for external Z3ENV.
Z3EADR:
        DW  Z3ENV                      ; Address of external
START:                                 ; Z3ENV descriptor.
        LD  HL,(Z3EADR)                ; Point to external
                                       ; descriptor.
        ELSE
; Conditional assembly still functioning. If
; external Z3ENV is not present, one must be
; installed in utility itself by calling macros
; from Z3BASE.LIB and SYSENV.LIB during assembly.
        MACLIB     Z3BASE.LIB          ; Note extended
        MACLIB     SYSENV.LIB          ; Intel format.
Z3EADR:
        JP         START               ; SYSENV is a macro used
        SYSENV                         ; to equip utility with
                                       ; internal descriptor.
START:
        LD  HL,Z3EADR                  ; Point to internal
                                       ; descriptor.
        ENDIF                          ; End of ELSE condition.
        PUSH  HL                       ; Move Z3ENV pointer to
        POP   IX                       ; Index Register X.
        CALL  Z3INIT                   ; Initializes Z3ENV.
        LD    HL,0                     ; Code continues.
        . . .
```

Table 2: *Excerpt from FINDF24.Z80, a fast find utility written by Richard Conn. All comments are mine.*

ECHELON'S Z-SYSTEM
(continued from page 43)

utility with macro options), and VMENU (an interactive shell of complex user options) are just a few of the many Z-System utilities that rely on flow control and multiple command processing for their power.

One of CP/M's strengths is its input/output redirection capability that uses the *STAT* command to revise the I/O byte at location 0003H. Z-System permits you to retain the CP/M device-assignment procedures or to use its own more efficient design based on the I/O drivers loaded as Input/Output Packages (IOPs). The Z-System approach to device assignment places any number of device drivers into memory, but only when needed. In contrast, the CP/M method restricts the number and types of drivers and requires them to be permanent residents of scarce CBIOS space, waiting to be needed. Like the other system segments, the IOP structure is supported by a variety of Z-System utilities such as DEVICE and RECORD, which select and activate or deactivate particular drivers. Users with elaborate hardware configurations can benefit most from the optional ZCPR3 IOP design.

Z-System's Languages

One feature of Z-System that has caused me some inconvenience with both ZCPR3 and its predecessor, ZCPR2, is the choice of mnemonics used in its source codes. To develop ZCPR3, Conn and the other designers of Z-System used Z80 assembly-language macro libraries, such as SYSLIB3, Z3LIB, and VLIB, which were written originally in extended Intel macros rather than "pure" Zilog mnemonics. Presumably, they wanted to make their source code accessible to the widest range of macro assemblers, including Digital Research's MAC and RMAC packages.

Until the past year, the source programs for most Z-System utilities, as well as the libraries themselves, were written only in extended Intel, with its bulky 8080 macros substituting for more compact and readable Z80 instructions. I prefer to code programs in Zilog mnemonics, and it is sometimes difficult to translate extended Intel into Zilog, particularly when the Intel macros have the same names as Zilog instructions. My solution has been to have several different assemblers in different named directories on the same disk. An interactive menu switches back and forth as needed.

Echelon is correcting this problem (if it is a problem for you) in its later software versions and now supplies source code in Zilog mnemonics as well as extended Intel. Zilog mnemonics are becoming more common in Echelon's Z-Tools collection of advanced system development software, much of which is now being sold with Zilog and HD64180 mnemonic patches to permit instant conversions and translations. Judging from recent issues of the *Z-News* newsletter, the Z-System community is tending to favor the newer 8-bit "superchips" such as the Hitachi HD64180 and the somewhat mythical Zilog Z800, whose instruction sets are upward compatible with Zilog mnemonics.

Advantage of Z80 Code

Z80 optimization of CP/M's 8080 code in the CCP and BDOS is one of the major strengths of Z-System. By using

more compact Z80 codes for relative and conditional jumps, block transfers, direct loading of double registers, and double-register arithmetic, the Z-System design team have compressed most CCP and BDOS routines tightly enough to add important features. The most common Z80 optimizations in Z-System are the familiar relative and conditional jumps, such as the common substitution of Zilog's *DJNZ LOOP* for the Intel *DEC B-JP NZ,LOOP* pair to control loops, a saving of only 2 bytes and either one or four clock cycles on each pass. Much more impressive optimizations are scattered throughout Z-System and its utilities, however.

In the ZRDOS *CHECKSUM* routine shown in the first part of Table 3, page 46, for example, Dennis Wright (not to be confused with Z-Com developer Joseph Wright) substitutes a single 2-byte, 15-cycle Zilog arithmetic instruction for a cumbersome 10-byte, 51-cycle Intel subroutine (*SUBDH* in the second part of Table 3) to subtract the contents of the *DE* register pair from those of the *HL* register pair. Note also how Wright's direct loading of the *DREC* parameter into registers *DE* eliminates the Intel *XCHG* instruction. Wright's Zilog substitutions in this tiny code sample alone save 7 bytes and 32 clock cycles each time they are used.

The powerful Z80 block-compare and -transfer instructions (*LDI, LDIR, CPI,* and *CPIR*) are underused in Conn's subroutine libraries and in the Z-System code produced from them. Wright, however, combines *LDIR* with direct loads of double registers to load the disk parameter block in the ZRDOS *SELECTDISK* routine with only 11 bytes and 52 clock cycles of code. Digital Research's comparable Intel code requires twice as many bytes and cycles to do the same thing. Wright's use of the Zilog *LDIR* instruction saves a total of 11 bytes and 46 clock cycles, as you can see by comparing the first and second parts of Table 4, page 46.

The most effective Z-System optimizations have less to do with Z80 features than with sound programming practices. In comparing disassembled source code for BDOS and ZRDOS, I noticed that most of Wright's improvements involved enhanced logic and the elimination of redundancy. By skillful sequencing of subroutines, he manages to make the flow between calling programs and ZRDOS system calls smoother and more efficient. He uses in-line code in preference to subroutines yet avoids redundancy by elegant relative loops.

The additional space harvested from his thoughtful assembly-language programming allows Wright to add several valuable routines and even four new system calls to ZRDOS. The major alterations in the CP/M 2.2 BDOS eliminate the nagging warm-boot requirement each time a disk is changed and make it easier to use the read-only disk status feature. ZRDOS also permits wheel protection of individual files and file archiving. Two other useful new system calls are included to add a warm-boot trap option so that users can customize error messages by diverting jumps to location 0000H.

Following the release of ZRDOS, several new utilities have appeared that take advantage of these enhancements, and the list is growing. Most Z-System utilities can work just as well under the CP/M BDOS, but newer ones such as AC (archive copy) and VIEW require ZRDOS instead. Echelon ad-

Excerpt from ZRDOS CHECKSUM routine. Code produces 11 bytes and 60 or 66 clock cycles.

```
        . . .
CHECKSUM:
        LD      HL,(DREC)               ; dir record into HL
        LD      DE,(CHKSIZ)             ; chksum vector to DE
        XOR     A                       ; clear A
        SBC     HL,DE                   ; compare by subtract
        RET     NC                      ; result placed in HL
                                        ; ret from CHECKSUM
                                        ; if DREC > CHKSIZ
        . . .                           ; else continue
```

Equivalent excerpt from CP/M 2.2 CHECKSUM routine. Code produces 18 bytes and 92 or 98 clock cycles.

```
        . . .
CHECKSUM:
        LHLD    DREC                    ; Dir record into HL.
        XCHG                            ; Shift to DE.
        LHLD    CHKSIZ                  ; Checksum vector to HL.
        CALL    SUBDH -                 ; Compare by subtract.
        RNC                             ; Return from CHECKSUM if
                                        ; DREC > CHKSIZ else
        . . .                           ; continue with CHECKSUM.
SUBDH:
        MOV     A,E                     ; Must move both bytes
        SUB     L                       ; of directory record into
        MOV     L,A                     ; register A for double
        MOV     A,D                     ; register subtract with
        SBB     H                       ; carry flag set—diff
        MOV     H,A                     ; is loaded in HL.
        RET     ; Return from SUBDH.
```

Table 3: *ZRDOS-BDOS code comparisons. Comments and labels are from my disassembly.*

Excerpt from CP/M 2.2 BDOS SELECTDISK routine. Code produces 22 bytes and 98 or 102 clock cycles.

```
MOVE:              ; Necessary in Intel.
        INR     C                       ; Set up for loop.
MOVE0:
        DEC     C                       ; Get zero if there.
        RZ                              ; End loop if empty.
        LDAX    D                       ; Source byte to A.
        MOV     M,A                     ; A to destination DPB.
        INX     D                       ; Next source byte.
        INX     H                       ; Next destination byte.
        JMP     MOVE0                   ; Loop until C=0.
SELECTDISK:
        . . .                           ; This subroutine fills
                                        ; the 16-byte disk
                                        ; parameter block.
        LHLD    DPBADDR                 ; HL points to source.
        XCHG                            ; Switch source to DE.
        LXI     H,SECTPT                ; HL=destination DPB.
        MVI     C,DPBLIST               ; C=size of DPB.
        CALL    MOVE                    ; Move it.
        . . .                           ; SELECTDISK continues.
```

Equivalent excerpt from ZRDOS SELECTDISK routine. Code produces 11 bytes and 52 or 57 clock cycles.

```
SELECTDISK:
        . . .
        LD HL,(DPBADDR)                 ; Source to HL.
        LD DE,SECTPT                    ; Destination to DE.
        LD BC,0FH                       ; DPB size=16 bytes.
        LDIR                            ; Move until done.
        . . .                           ; Resume SELECTDISK.
```

Table 4: *ZRDOS-BDOS code comparisons*

ECHELON'S Z-SYSTEM
(continued from page 45)

vertises a more powerful version, ZRDOS3, designed specifically for the Hitachi HD64180 and Zilog Z800 boards. I have not had an opportunity to study ZRDOS3, but Echelon says it adds around 50 new system calls to handle such advanced 8-bit features as multitasking, full-track disk buffering, and routines for addressing larger RAMs.

Documentation

Conn's book *ZCPR3: The Manual* is the installation bible and technical reference for ZCPR3 and its utilities, particularly if you're installing the manual-install hacker version. Echelon also distributes *Z-System User's Guide* by Richard Jacobson and Bruce Morgen, which is more appropriate for novice users and others who are content with the Z-Com version. Many other well-written hard-copy manuals on special topics and tools are also available from Echelon.

Z-Com comes with more than 400K of on-line modular help files on all aspects of the system, and each utility has built-in help screens for immediate access. Echelon and Z-Node operators are never stingy with source code. A few proprietary items, such as ZRDOS and some of the Z-Tools, are distributed only in object-code files, but nearly everything else is available in compact assembly-language libraries with professional documentation and helpful programming suggestions.

Conclusions

The major advantages of Z-System are Z80 code optimizations, enhanced transportability, powerful new user options, and compatibility with CP/M 2.2. I've never encountered such elegant and efficient code in any other operating system that I've worked with.

My favorite components are the RAM-resident utilities, internal flow control with nested logic, aliases and multiple commands, named directories, and customized menu shells, but these are just a few of Z-System's advanced features. Combined with fast new Hitachi HD64180 boards, Echelon's ZCPR3 and ZRDOS leave very little to be desired in an 8-bit operating system.

Bibliography

Conn, Richard. *ZCPR3: The Libraries*. Los Altos, Calif.: Echelon, 1986. A manual and guide for Z80 system developers using the SYSLIB, Z3LIB, and VLIB Z80 subroutine libraries.

Conn, Richard. *ZCPR3: The Manual*. New York: Zoetrope, 1985. The most comprehensive and technical reference for ZCPR3 and its utilities.

Jacobson, Richard, and Morgen, Bruce. *Z-System User's Guide*. Los Altos, Calif.: Echelon, 1986. Ideal for beginning Z-System users, especially those with the Z-Com auto-install version.

These books and other Z-System materials are available directly from Echelon. For further information and current prices, write or call Echelon Inc., 885 N. San Antonio Rd., Los Altos, CA 94022; (415) 948-3820.

DDJ

Vote for your favorite feature/article.
Circle Reader Service **No. 4.**

Series 32000 Cross Assembler

by Richard Rodman

The 32000 processor features generalized addressing modes available in almost all instructions.

The National Semiconductor series 32000 microprocessor line includes the 32-bit 32032 and the 16-bit 32016 (formerly called 16032) microprocessors. As part of a project to build a board using a 32032, I wrote an assembler in Software Toolworks' C/80; adaptation to any other variant of C should be easy.

Although most people lump the 68000 and the 32016 together, these processors are radically different. The differences have been summed up as "the 68000 is PDP-11-like, whereas the 32000 is VAX-like." The 32000 includes bit-field, translate, procedure enter/return, and other high-level instructions in its instruction set.

Basic Program Design

This program works in a brute-force fashion, but it is easy to understand, modify, and debug. Each instruction's binary equivalent is stored in a string, with *x*s where operands need to be inserted. A string matcher, *match( )*, matches the opcodes against lines in the source file, keeping matches to wildcards in the buffer *ambig_buffer*. Each opcode has an option character, *opopt*, associated with it that controls special-case logic for some instructions. The data is output in Intel absolute hex format. Table 1, page 49, shows the definitions for the *opopt* characters and the instruction table format. Table 2, page 49, shows some examples of instruction formats defined using the structure in Table 1.

The 32000 processor, although allowing absolute addressing, features generalized addressing modes available in almost all instructions. Two's-complement offsets can be used in three different sizes—7, 14, or 30 bits long—as needed. Because these offsets could refer to areas not yet defined, and the length of the code varies with the offset, three passes are necessary. The first pass gets a coarse value of all symbols, the second pass then makes the variable offsets the right length and corrects the symbol values, and the third pass actually generates the code. After the first pass, the symbol table is sorted; then in the second and third passes, a binary search is used to find entries more quickly.

Richard Rodman, 1923 Anderson Rd., Falls Church, VA 22043

Assembler Syntax

• Symbols—This assembler limits line length to 128 characters; symbols can be up to a whole line long. Labels must be followed by a colon and can not be reused. The colon must be omitted on equates. Values assigned with *equ* can be redefined, however.

• Pseudo-ops—*org* must be followed by a value. Although the 32000 does not require word alignment of code or data, it does make some operations faster, so an *even* pseudo-op is provided to force the code address to an even boundary.

Define byte, word, double (*db*, *dw*, *dd*) must be only one value per line. Currently, character-string constants are not supported.

Numeric constants must begin with a digit. Default radix is decimal, or the value can be followed with an *h*, *q*, or *b* for hexadecimal, octal, or binary, respectively. The code address is known as ".", and the assembly address (which may be different) as "..".

• Opcodes—All 32000 opcodes are supported. The assembly instructions must conform to the *NS16000 Instruction Set Reference Manual*—for example, arguments to the *SAVE* instruction must be enclosed in square brackets. You can include multiple instructions on a line as long as all operands to each instruction are provided.

• Comments—Comments begin with a semicolon (;) and continue to the end of the line. Some programmers have the bad habit of omitting the beginning * or ;. That won't work here.

• Assembly-time arithmetic—Only "+", "*", "−", and "/" are supported. A look at the listing shows it would be trivial to add more operators, however. Formulas are allowed anywhere a value is required, but they must be enclosed in parentheses. Within parentheses, values must be separated from operators with spaces. This is because the program uses the spaces to tell where words end, and math operators are considered words. Spaces are not needed between the parentheses and the words enclosed. Note the spaces around + and /. An example will best illustrate:

```
((FEN + 1) + (GUG / 3))
```

Commas also separate words; in fact, commas and spaces are interchangeable, although human readers may consider commas out of place in some instances.

• Listing—The assembler produces a listing on the final pass. This listing is sent to the screen but can be redirected into a file or to the printer. It is a traditional listing, with address, bytes of code or data, and opcodes and

```
#define MAXOP 149
/* the opcode binary value should be a string of bits e.g. 0111xxxxx000b
the opcode opopt character is used to specify special operands, etc. */

/* opopts used here for the 32000 are:
        blank       nothing special
        a           gen
        b           gen short
        c           gen gen
        d           00000 short
        e           gen gen reg
        f           reglist save/enter
        g           reglist restore/exit
        h           00000 gen (sfsr)
        i           inss/exts
        j           movs/skps/cmps
        k           setcfg
        l           procreg, gen for lpr/spr
        m           index (operand order)
        n           ret/rett — postbyte
        o           movm
        p           cxp (disp after instruction) */

struct {
        char *onam;         /* opcode name */
        int ocnt;           /* operand count, negative if PC-relative */
        char *obin;         /* opcode binary value */
        char oopt;          /* opcode opopt char */
}
```

Table 1: *Definitions of* opopt *characters*

```
"bsr",      -1,   "02h",                              ' ',
"save",      1,   "62h",                              'f',
"svc",       0,   "0e2h",                             ' ',
"bne",      -1,   "1ah",                              'b',
"addq?",     2,   "xxxxxxxxx00011iib",                'e',
"sgt?",      1,   "xxxxx011001111iib",                'a',
"jump",      1,   "xxxxx01001111111b",                'a',
"jsr",       1,   "xxxxx11001111111b",                'a',
"addl",      2,   "xxxxxxxxxx00000010111110b",        'c',
"mulf",      2,   "xxxxxxxxxx11000110111110b",        'c',
"and?",      2,   "xxxxxxxxxx1010iib",                'c',
"not?",      2,   "xxxxxxxxxx1001ii01001110b",        'c',
```

Table 2: *Selected instruction formats from the opcode table*

?	unknown item—syntax error
x	unimplemented instruction (bad instruction database)
l	no length modifier (bad instruction database) or expression too complex
e	address extensions missing
p	illegal register *lpr/spr*
[	brackets required
v	syntax error in value
o	unknown arithmetic operator
u	undefined symbol

Table 3: *Error messages*

comments on the right.

Table 3, below, shows the error messages produced by the assembler.

Future Enhancements

Unless I get some 32000 hardware to play with, it's unlikely I'll work on this program further. If you'd like to work on it, however, some items on your list should be:

1. Multivalue *db/dw/dd* and character-string constants.
2. Global/external object format and linker. The 32000 instructions are already relocatable; any absolute values that would be present would presumably be entry points or I/O addresses. In fact, even the global/external isn't really necessary because of the *cxp/rxp* instructions.
3. *Cseg/dseg* pseudo-ops.

If you send your changes to me, I'll be happy to make them available to others. Anyone wanting a copy of the source code may send me $8 for materials and effort. Please specify 8-inch CP/M, 5¼-inch PC, or other (inquire) or 3½-inch Atari ST.

For those lucky people who are in a position to make use of this program, why not let readers know what you're doing? Is the 32000 really the programmer's dream some say it is? And for those who are in a position to do so, how about some inexpensive 32000 hardware—a single-board computer perhaps—so people can get a hands-on feel for what the processor can do?

Even if you don't have a 32000 processor to play with, you may be able to make use of routines from this program. The style exemplifies my belief that C should be written to be readable both by computers and by humans. Cryptic C is bad C.

DDJ

Vote for your favorite feature/article.
Circle Reader Service **No. 5.**

The listing for this article is presented in a machine-readable form—Softstrips produced by Cauzin Systems. The strips begin on page 83. The text of the listing is available for downloading in the DDJ Electronic Edition on CompuServe. A disk with this listing and others is also available—see the ad on page 115. The text of the listing will be published next month.

Listing One (Text begins on page 16.)

```
/*
Listing 1      Scheduling Algorithm
(C) Copyright 1986 Ken Berry.
All rights reserved.
Copies may be made for non-commercial, private use only.
*/

#define _F 0                /* false */
#define _T 1                /* true */
#define _E -1               /* error */

#define _NULL 0             /* null pointer */

typedef char pointer;       /* pointer type */
typedef char logical;       /* logical type */
typedef unsigned selector;  /* 8086 selector type */

struct sys_parm             /* register storage block for 8086 interface */
{
  union {unsigned sys_rax; struct {char sys_ral, sys_rah;} sys_byt;} sys_ra;
  union {unsigned sys_rbx; struct {char sys_rbl, sys_rbh;} sys_byt;} sys_rb;
  union {unsigned sys_rcx; struct {char sys_rcl, sys_rch;} sys_byt;} sys_rc;
  union {unsigned sys_rdx; struct {char sys_rdl, sys_rdh;} sys_byt;} sys_rd;
#define sys_ax sys_ra.sys_rax
#define sys_al sys_ra.sys_byt.sys_ral
#define sys_ah sys_ra.sys_byt.sys_rah
#define sys_bx sys_rb.sys_rbx
#define sys_bl sys_rb.sys_byt.sys_rbl
#define sys_bh sys_rb.sys_byt.sys_rbh
#define sys_cx sys_rc.sys_rcx
#define sys_cl sys_rc.sys_byt.sys_rcl
#define sys_ch sys_rc.sys_byt.sys_rch
#define sys_dx sys_rd.sys_rdx
#define sys_dl sys_rd.sys_byt.sys_rdl
#define sys_dh sys_rd.sys_byt.sys_rdh
  unsigned sys_bp;          /* base pointer */
  unsigned sys_si;          /* source index */
  unsigned sys_di;          /* destination index */
  unsigned sys_sp;          /* stack pointer */
  unsigned sys_cs;          /* code segment */
  unsigned sys_ds;          /* data segment */
  unsigned sys_ss;          /* stack segment */
  unsigned sys_es;          /* extra segment */
  unsigned sys_pf;          /* 80286 processor flags */
#define SYS_OF 0x0800           /* overflow flag- 1: lost significance */
#define SYS_DF 0x0400           /* direction flag- 1: strings auto-decrement */
#define SYS_IF 0x0200           /* interrupt flag- 1: enable interrupts */
#define SYS_TF 0x0100           /* trap flag- 1: interrupt every instruction */
#define SYS_SF 0x0080           /* sign flag- 1: result negative */
#define SYS_ZF 0x0040           /* zero flag- 1: result 0 */
#define SYS_AF 0x0010           /* auxiliary carry flag- 1: carry from bit 3 */
#define SYS_PF 0x0004           /* parity flag- 1: even number of 1's */
#define SYS_CF 0x0001           /* carry flag- 1: carry from bit 8 or 16 */
  unsigned sys_sw;          /* status word */
#define SYS_TS 0x0008           /* task switch */
#define SYS_EM 0x0004           /* processor extension emulation */
#define SYS_MP 0x0002           /* monitor processor extension */
#define SYS_PE 0x0001           /* protection enable */
  unsigned sys_ip;          /* instruction pointer */
  unsigned sys_res;         /* unused */
};

struct t_xstck
{
  unsigned t_xbase;         /* application stack base (overflow detection) */
  unsigned t_xes;           /* es */
  unsigned t_xbp;           /* bp */
  unsigned t_xdi;           /* di */
  unsigned t_xsi;           /* si */
  unsigned t_xdx;           /* dx */
  unsigned t_xcx;           /* cx */
  unsigned t_xbx;           /* bx */
  unsigned t_xax;           /* ax */
  unsigned t_xds;           /* ds */
  unsigned t_xip;           /* ip */
  unsigned t_xcs;           /* cs */
  unsigned t_xpf;           /* pf */
  unsigned t_retip;         /* return address */
};

struct t_task
{
  char t_type;              /* task type */
#define T_X 0x80            /* execute queue */
#define T_W 0x40            /* wait queue */
#define T_P 0x20            /* priority queue */
#define T_SW 0x10           /* secondary wait queue */
#define T_ATASK 0x01        /* abreviated task */
  unsigned t_wttk;          /* wait tick count */
  unsigned t_cls;           /* priority queue index */
  struct t_task *t_pqtsk,*t_nqtsk; /* queue linkage */
  struct t_task *t_ratsk,*t_pstsk,*t_nstsk,*t_fdtsk,*t_ldtsk; /* family */
  struct sys_parm t_ps;     /* processor status */
  unsigned t_xtm0;          /* execution time accumulator */
  unsigned t_xtm1;
  unsigned t_xtm2;
  pointer *t_axstk;         /* execution stack pointer */
};

extern pointer *sys_task; /* current task control table pointer */
#define _tsk ( ( struct t_task * ) sys_task ) /* task control table ref */

#define T_SCLS 4            /* number of scheduling classes */

struct t_scls               /* scheduling class queue */
{
  unsigned t_sfrq;          /* scheduling frequency */
  int t_sct;                /* queue length */
```

```
  struct t_task *t_fqtsk,*t_lqtsk; /* queue header */
};

struct t_schd               /* scheduling control table */
{
  int t_xct;                /* execution queue length */
  struct t_task *t_fxtsk, *t_lxtsk; /* execution queue header */
  int t_wct;                /* wait queue length */
  struct t_task *t_fwtsk, *t_lwtsk; /* wait queue header */
  int t_swct;               /* secondary wait queue length */
  struct t_task *t_fswtsk, *t_lswtsk; /* secondary wait queue header */
  int t_sclsl;              /* scheduling class index limit */
  struct t_scls **t_sclsp; /* scheduling class array pointer */
};

extern pointer *sys_tsch; /* task scheduling control table pointer */
#define _tschd ( ( struct t_schd * ) sys_tsch ) /* quick pointer */

/*
t__krnl                     /* security kernel */
*/
t__krnl()
/*
This is the security kernel.  It never returns, being the most trusted
software in the system. The current contents in t__crtss and t__crtsp
are used to set the stack for when the current task is resumed. */
{
  extern logical t_astrm; /* tick termination flag */
  extern selector t__crtss; /* current task ss storage */
  extern pointer *t__crtsp; /* current task sp storage */
  extern unsigned tmr_tkct; /* tick clock */
  int xtskct;               /* task queue count (at entry) */
  int ttc;                  /* task termination code */
  _tsk -> t_ps.sys_ss = t__crtss; /* set current task stack */
  _tsk -> t_ps.sys_sp = t__crtsp;
  while(_T)                 /* find executable task */
  {
    xtskct = _tschd -> t_xct; /* save task count */
    if ( t_astrm ) t__wtst( tmr_tkct ); /* process wait tasks */
    if ( xtskct == 0 ) t__sch(); /* schedule application tasks if necessary */
    sys_task = _tschd -> t_fxtsk; /* set next task address */
    if ( sys_task != _NULL )  /* test for executable task available */
    {
      _tschd -> t_xct--; /* decrement executing task count */
      _tschd -> t_fxtsk = _tsk -> t_nqtsk; /* delink task */
      if ( _tschd -> t_fxtsk == _NULL )
        _tschd -> t_lxtsk = _NULL;
      else _tschd -> t_fxtsk -> t_pqtsk = _NULL;
      _tsk -> t_type &= ~T_X; /* Indicate task not in execution queue */
      ttc = t__xtsk( sys_task ); /* execute application task */
      if ( !sys_task ) continue; /* test for task terminated */
      if ( ttc < 0 ) t__inxq(); /* insert task into execution queue */
      else if ( ttc == 0 ) t__inpq(); /* insert task into priority queue */
      else t__inwq( ttc ); /* insert into wait queue */
    }
  }
}

/*
t__wtst                     test waiting tasks
*/
t__wtst( tc)
unsigned tc;
/*
The wait queue is traversed.  All tasks with a wait value of tc are executed.
_F is always returned. */
{
  while(_T)                 /* traverse wait queue */
  {
    sys_task = _tschd -> t_fwtsk; /* set current task pointer */
    if ( !sys_task ) break; /* test for no waiting tasks */
    _tsk -> t_type &= ~T_W; /* remove task from wait queue */
    _tsk -> t_type |= T_X; /* indicate task in execution queue */
    if ( _tsk -> t_wttk > tc ) break; /* test for effective end of list */
    -- _tschd -> t_wct;   /* decrement waiting task  count */
    _tschd -> t_fwtsk = _tsk -> t_nqtsk; /* delink from wait queue */
    if ( _tsk -> t_nqtsk == _NULL )
      _tschd -> t_lwtsk = _NULL;
    else _tsk -> t_nqtsk -> t_pqtsk = _NULL;
    _tsk -> t_pqtsk = _NULL; /* insert at top of execution queue */
    _tsk -> t_nqtsk = _tschd -> t_fxtsk;
    _tschd -> t_fxtsk = sys_task;
    ++_tschd -> t_xct;   /* increment executable task count */
    if ( _tschd -> t_lxtsk == _NULL )
      _tschd -> t_lxtsk = sys_task;
    else _tsk -> t_nqtsk -> t_pqtsk = sys_task;
  }
  return _F;                /* return */
}

/*
t__sch                      schedule task
*/
t__sch()
/*
This function searches the priority queues and links tasks ready for execution
into the execution queue. The return is always _F. */
{
  struct t_scls **a;        /* priority queue pointer array pointer */
  struct t_scls *q;         /* priority queue pointer */
  int i,j;                  /* iteration variables */
  a = _tschd -> t_sclsp; /* set pointer array address */
/*  while(_T)                 /* nonterminating task */
  { */
    for ( i = 0; i < _tschd -> t_sclsl; ++i ) /* traverse queues */
    {
      q = a[i];             /* set priority queue pointer */
      for ( j = 0; j++<q -> t_sfrq; ) /* schedule tasks from priority queue */
    {
      if ( q -> t_fqtsk == _NULL ) break; /* test for queue empty */
      if ( _tschd -> t_lxtsk ) /* link to end of execution queue */
        _tschd -> t_lxtsk -> t_nqtsk = q -> t_fqtsk;
```

(continued on next page)

MULTITASKING KERNEL

Listing One (Listing continued, text begins on page 16.)

```
        else _tschd -> t_fxtsk = q -> t_fqtsk;
        q -> t_fqtsk -> t_type &= ~T_P; /* indicate not in priority queue */
        q -> t_fqtsk -> t_pqtsk = _tschd -> t_lxtsk;
        _tschd -> t_lxtsk = q -> t_fqtsk;
        q -> t_fqtsk = q -> t_fqtsk -> t_nqtsk; /* update queue header */
        _tschd -> t_lxtsk -> t_nqtsk = _NULL;
        if ( q -> t_fqtsk == _NULL )
          q -> t_lqtsk = _NULL;
        else q -> t_fqtsk -> t_pqtsk = _NULL;
        q -> t_sct --;    /* decrement queue count */
        ++ _tschd -> t_xct; /* increment execution queue length */
        _tschd -> t_lxtsk -> t_type |= T_X; /* ind. task in execution queue */
      }
    }
/*    t_rels();              /* return to bottom of execution queue */
  }*/
  return _F;               /* return */
}

/*
t__inxq                    insert task into execution queue
*/
t__inxq()
/*
The current task is inserted into the execution queue. _F is always returned.
*/
{
  _tsk -> t_wttk = 0;     /* indicate not waiting for system tick */
  if ( _tschd -> t_lxtsk == _NULL ) /* test for execution queue empty */
  {
    _tschd -> t_fxtsk = sys_task; /* insert in empty queue */
    _tsk -> t_pqtsk = _NULL;
  }
  else                     /* execution queue not empty */
  {
    _tschd -> t_lxtsk -> t_nqtsk = sys_task; /* insert at end of queue */
    _tsk -> t_pqtsk = _tschd -> t_lxtsk;
  }
  _tsk -> t_nqtsk = _NULL; /* new task at end of list */
  _tschd -> t_lxtsk = sys_task;
  _tschd -> t_xct++;       /* increment executable task count */
  _tschd -> t_lxtsk -> t_type |= T_X; /* indicate task in execution queue */
  return _F;               /* return */
}

/*
t__secw                    process secondary wait queue
*/
t__secw()
/*
This program executes every 64K system ticks. It moves the secondary
wait queue to the primary wait queue and changes the type of the waiting
tasks. */
{
  struct t_task *tsk;      /* task control table pointer */
  char swtflg;             /* system state flag */
  while(_T)                /* nonterminating task */
  {
    t__syntr( &swtflg ); /* enter system state */
    for ( tsk = _tschd -> t_fwtsk; tsk; tsk = tsk -> t_nqtsk ) /* traverse */
    {
      tsk -> t_type &= ~T_SW;
      tsk -> t_type |= T_W; /* change task type */
    }
    _tschd -> t_wct = _tschd -> t_swct; /* append secondary wait queue */
    _tschd -> t_fwtsk = _tschd -> t_fswtsk;
    _tschd -> t_lwtsk = _tschd -> t_lswtsk;
    _tschd -> t_fswtsk = _tschd -> t_lswtsk = _NULL; /* empty sec. queue */
    _tschd -> t_swct = 0;
    t__inwq( 0xFFFF - tmr_tkct ); /* insert self into wait queue at end */
    sys_task = _NULL;      /* remove task from kernel control */
    t_term();              /* suspend execution */
  }
}

/*
t__inwq                    insert task into wait queue
*/
t__inwq(tc)
unsigned tc;
/*
The  current task is inserted into the wait queue.  tc is the number of system
ticks that the task is to wait. _F is always returned. */
{
  extern unsigned tmr_tkct; /* tick clock */
  unsigned crtk;           /* current tick */
  crtk = tmr_tkct;         /* set current system tick */
  _tsk -> t_wttk = tc + crtk; /* compute reactivation time */
  if ( _tsk -> t_wttk >= crtk ) /* test for task in wait queue */
  { t__inwt( &_tschd -> t_wct ); /* insert in wait queue */
    _tsk -> t_type |= T_W;
  }else                    /* task in secondary wait queue */
  { t__inwt( &_tschd -> t_swct ); /* insert in secondary wait queue */
    _tsk -> t_type |= T_SW;
  }return _F;              /* indicate task inserted */
}

/*
t__inwt                    insert into wait or secondary wait queue
*/
struct t_wtq
{
  int wct;                 /* wait queue length */
  struct t_task *frs,*lst; /* queue header */
};
t__inwt( w )
struct t_wtq *w;
```

```
/*
The t_wtq structure is implicitly contained in the scheduling control table
(t_schd structure).  The current task is inserted into the queue. _F is always
returned. */
{
  struct t_task *p;        /* task pointer */
  unsigned tc;             /* reactivation time */
  tc = _tsk -> t_wttk;     /* set reactivation time */
  ++w -> wct;              /* increment queue length */
  for ( p = w -> frs; p; p = p -> t_nqtsk ) /* traverse queue */
  {
    if ( tc < p -> t_wttk ) /* test for task earlier */
    {
      _tsk -> t_nqtsk = p; /* insert within queue */
      _tsk -> t_pqtsk = p -> t_pqtsk;
      p -> t_pqtsk = sys_task;
      if ( ( p = _tsk -> t_pqtsk ) ) p -> t_nqtsk = sys_task;
      else w -> frs = sys_task;
      return _F;           /* indicate task inserted */
    }
  }
  if ( ( p = w -> lst ) ) /* test for wait queue not empty */
  {
    p -> t_nqtsk = w -> lst = sys_task; /* insert at end of queue */
    _tsk -> t_pqtsk = p;
    _tsk -> t_nqtsk = _NULL;
  }
  else                     /* wait queue empty */
  {
    w -> frs = w -> lst = sys_task; /* initialize wait queue */
    _tsk -> t_nqtsk = _tsk -> t_pqtsk = _NULL;
  }
  return _F;               /* indicate task inserted */
}

/*
t__inpq                    insert into priority queue
*/
t__inpq()
/*
The current task is inserted into its priority queue. _F is always returned.
*/
{
  struct t_scls *q;        /* priority queue pointer */
  _tsk -> t_wttk = 0;      /* indicate not waiting for tick */
  q = _tschd -> t_sclsp[ _tsk -> t_cls ]; /* set priority queue address */
  _tsk -> t_pqtsk = q -> t_lqtsk; /* link task into priority queue */
  _tsk -> t_nqtsk = _NULL;
  if ( q -> t_lqtsk == _NULL ) q -> t_fqtsk = sys_task;
  else q -> t_lqtsk -> t_nqtsk = sys_task;
  q -> t_lqtsk = sys_task;
  ++q -> t_sct;            /* increment queue length */
  _tsk -> t_type |= T_P;   /* indicate task in priority queue */
  return _F;               /* return */
}

/*
t__xtsk                    execute task
*/
t__xtsk( t )
struct t_task *t;
/*
Task t is executed. The returned value is the termination code.
*/
{
  extern unsigned t_mnxtm; /* minimum execution time */
  extern unsigned t_syxtm[]; /* system pseudo time accumulator */
  extern logical t_astrm; /* application termination flag */
  int ttc;                 /* return value storage */
  unsigned atm;            /* accumulated time */
  unsigned rtm;            /* reference time */
  int xtm;                 /* execution time */
  atm = 0;                 /* initialize accumulated execution time */
  while(_T)                /* execute task */
  {
    rtm = t -> t_xtm0;     /* set reference time */
    t_rtmark( &t -> t_xtm0 ); /* accumulate pseudo time */
    ttc = t__dspap( t -> t_ps.sys_ss, t -> t_ps.sys_sp ); /* execute task */
    t -> t_ps.sys_ss = t__crtss; /* store ss */
    t -> t_ps.sys_sp = t__crtsp; /* store sp */
    t_rtmark( &t_syxtm ); /* accumulate pseudo time */
    if ( ( ttc != 0 ) || !t_astrm ) break; /* test for not tick termination */
    xtm = t -> t_xtm0 - rtm; /* compute execution time */
    if ( xtm < rtm ) xtm = -xtm;
    atm += xtm;            /* accumulate execution time */
    if ( atm >= t_mnxtm ) break; /* test for minimum time satisfied */
  }
  return ttc;              /* return */
}
/*
t__init                    initialize task system
*/
t__init()
/*
This function initializes the task system. _F is the normal return. _E is
returned if the system cannot be initialized. */
{
#define WSTK 252           /* t_wqupd stack size */
  extern struct sys_parm sys_stat; /* initial processor status */
  extern struct t_task *t_wqupd; /* secondary wait queue update task */
  extern selector sys_dgrp; /* data segment selector storage */
  extern char *sys_ssbs; /* system stack pointer */
  extern unsigned sys_sssz; /* system stack length */
  extern char tmr_ilck;   /* tick service interlock */
  int t__secw();          /* wait queue update function */
  struct t_scls *cls;     /* priority queue pointer */
  struct t_scls **ary;    /* priority queue pointer array pointer */
  int i;                  /* iteration variable */
  char *s;                /* pointer */
  tmr__int();             /* initialize system tick clock */
  sys_task = sys_ssbs + sys_sssz; /* set main task control table pointer */
  _tsk -> t_xtm0 =        /* initialize execution time */
  _tsk -> t_xtm1 =
  _tsk -> t_xtm2 = 0;
```

(continued on next page)

MULTITASKING KERNEL

Listing One (Listing continued, text begins on page 16.)

```
  if( ( sys_tsch = mm_aloc( sizeof( struct t_schd ) ) ) == _NULL )goto err1;
  if( ( ary = mm_aloc( T_SCLS*( sizeof( struct t_scls )+2 ) ) )
     == _NULL )goto err2;
  _tsk -> t_pqtsk =         /* NULL linkage */
  _tsk -> t_nqtsk =
  _tsk -> t_ratsk =
  _tsk -> t_pstsk =
  _tsk -> t_nstsk =
  _tsk -> t_fdtsk =
  _tsk -> t_ldtsk = _NULL;
  _tsk -> t_cls = 0;        /* set priority class 0 */
  _tsk -> t_wttk = 0;       /* indicate not waiting */
  for( i = 0, s = &_tsk -> t_ps;
    i++<sizeof( struct sys_parm ); )*s = 0; /* clear t_ps */
  _tsk -> t_ps.sys_cs = sys_stat.sys_cs; /* set selectors */
  _tsk -> t_ps.sys_ds =
  _tsk -> t_ps.sys_es =
  _tsk -> t_ps.sys_ss = sys_dgrp;
  _tsk -> t_axstk = _NULL; /* NULL execution stack pointer */
  _tschd -> t_xct = 1;    /* set execution task count */
  _tschd -> t_fxtsk =     /* set execution queue */
  _tschd -> t_lxtsk = sys_task;
  _tschd -> t_wct = 0;    /* indicate idle wait queue */
  _tschd -> t_fwtsk =     /* set wait queue */
  _tschd -> t_lwtsk = _NULL;
  _tschd -> t_swct = 0;   /* indicate empty secondary wait queue */
  _tschd -> t_fswtsk =    /* NULL secondary wait queue */
  _tschd -> t_lswtsk = _NULL;
  _tschd -> t_sclsl = T_SCLS; /* set priority queue count */
  _tschd -> t_sclsp = ary; /* set priority queue pointer array address */
  cls = &ary[ T_SCLS ];   /* set first t_scls pointer */
  for( i = 0; i<T_SCLS; ++i, ++cls ) /* initialize priority queues */
  {
    ary[i] = cls;          /* set priority queue pointer */
    cls -> t_sfrq = 1;     /* set default frequency */
    cls -> t_sct = 0;      /* indicate empty queue */
    cls -> t_fqtsk =       /* NULL queue linkage */
    cls -> t_lqtsk = _NULL;
  }
  t_wqupd =                /* create task to update wait queue */
    t_crt( t__secw, 0, 0, WSTK, 0, 0, 0 );
  if( t_wqupd == _NULL )goto err3;
  t_wqupd -> t_wttk = 0xFFFF; /* update wait queue at wraparound time */
  t__dspsy();              /* dispatch system */
  tmr_ilck = 0x00;         /* enable tick service */
  return _F;               /* indicate task system initialized */
err3: mm_free( ary );      /* error nest */
err2: mm_free( sys_tsch );
err1: return _E;
}

/*
t__term
*/
t__term()
/*
The task system is terminated. All tasks and storage allocated by t__init are
released. The return is always _F. */
{
  extern char *sys_ssbs; /* system stack base */
  extern unsigned sys_sssz; /* system stack size */
  struct t_task *t;        /* t_task pointer */
  char trmflg;             /* system state flag storage */
  tmr__rst();              /* reset system tick clock */
  t__syntr( &trmflg );     /* enter system state */
  sys_task = sys_ssbs + sys_sssz; /* set original task address */
  while( ( t = _tsk -> t_fdtsk ) ) /* delete all created tasks */
    t_del( t, _F );
  mm_free( _tschd -> t_sclsp );
  mm_free( sys_tsch );
  return _F;               /* normal return */
}
/*
t_crt                      create task
*/
t_crt( xadr, pcnt, padr, ssiz, dsiz, sadr, prty )
pointer *xadr;
unsigned pcnt;
unsigned *padr;
unsigned ssiz, dsiz;
pointer *sadr;
unsigned prty;
/*
A  new task is created with execution priority prty.  Execution will begin  at
xadr.  pcnt parameters will be passed (on the new task stack).  The parameters
are  in  an array addressed by padr.  The new task will have a stack  of  ssiz
bytes  and a dynamic memory area of dsiz bytes.  dsiz may be zero to  indicate
that no dynamic memory is required.  sadr will recieve a termination code when
the  task  terminates. If sadr is _NULL, an abreviated task is created.  _F  is
returned if insufficient memory  is  available.  Otherwise the address of the
t_ftask table is returned. */
{
  extern int t__halt();  /* return address */
  struct t_task *tsk;    /* task control table pointer (t_task) */
  struct t_scls *pq;     /* priority queue pointer */
  struct t_xstck *sp;    /* execution stack pointer */
  pointer *ss;           /* stack start */
  unsigned *pr;          /* parameter pointer */
  unsigned ln;           /* task control table length */
  int i;                 /* iteration variable */
  char *s;               /* pointer */
  char *sptr;            /* execution stack pointer */
  logical crtflg;        /* system state flag storage */
  t__syntr( &crtflg );   /* enter system state */
  ln = sizeof( struct t_task );  /* allocate task control table */
  if( ( tsk = mm_aloc( ln ) ) == _NULL ) goto err1;
  ssiz += sizeof( struct t_xstck ); /* allocate stack */
```

(continued on page 56)

Now You Know Why BRIEF™ is BEST

"BRIEF is simple to learn and use and extremely sophisticated."

PC Magazine, July 1986

"A bona fide Undo..."

Steve McMahon, BYTE 3/85

As Mark Edwards describes in DR. DOBB'S JOURNAL (11/85), "BRIEF has an outstanding undo facility. The default configuration allows the last 30 editing commands to be undone. This number can be raised to a maximum of 300 commands. Until you reach this maximum or run out of RAM, every command you issue can be undone. So if you make ten changes and then realize that the first one was an error, you can undo all the changes back to the mistake... Needless to emphasize, this facility can save endless grief."

No other editor has this capability.

The Program Editor with the BEST Features

Since its introduction, BRIEF has been sweeping programmers off their feet. Why? Because BRIEF offers the features **MOST ASKED FOR** by professional programmers. In fact, BRIEF has just about every feature you've ever seen or imagined, including the ability to configure windows, keyboard assignments, and commands to **YOUR** preference. One reviewer (David Irwin, DATA BASED ADVISOR) put it most aptly, "(BRIEF)...is quite simply the best code editor I have seen."

Every Feature You Can Imagine

Compare these features with your editor (or any other for that matter).

- FAST
- Full UNDO (N Times)
- Edit Multiple Large Files
- Compiler-specific support, like auto indent, syntax check, compile within BRIEF, and template editing
- Exit to DOS inside BRIEF
- Uses all Available Memory
- Tutorial
- Repeat Keystroke Sequences
- 15 Minute Learning Time
- Windows (Tiled and Pop-up)
- Unlimited File Size –(even 2 Meg!)
- Reconfigurable Keyboard
- Context Sensitive Help
- Search for "regular expressions"
- Mnemonic Key Assignments
- Horizontal Scrolling
- Comprehensive Error Recovery
- A Complete Compiled Programmable and Readable Macro Language
- EGA and Large Display Support
- Adjustable line length up to 512

Program Editing YOUR Way

A typical program editor requires you to adjust your style of programming to its particular requirements – NOT SO WITH BRIEF. You can easily customize BRIEF to your way of doing things, making it a natural extension of your mind. For example, you can create ANY command and assign it to ANY key – even basic function keys such as cursor-control keys or the return key.

The Experts Agree

Reviewers at BYTE, INFOWORLD, DATA BASED ADVISOR, and DR. DOBB'S JOURNAL all came to the same conclusion – BRIEF IS BEST!

Further, of 20 top industry experts who were given BRIEF to test, 15 were so impressed they scrapped their existing editors!

NOT COPY PROTECTED

Solution Systems™

MONEY-BACK GUARANTEE
Try BRIEF ($195) for 30 days – If not satisfied get a full refund.
TO ORDER CALL (800-821-2492)

SOLUTION SYSTEMS, 335-D WASHINGTON ST., NORWELL, MA 02061, 617-659-1571

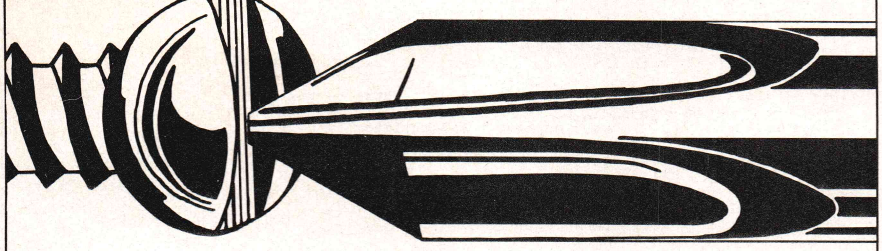
BRIEF is a trademark of UnderWare

Listing One (Listing continued, text begins on page 16.)

```
  if( ( ss = tsk -> t_axstk = mm_aloc( ssiz ) ) == _NULL )goto err2;
  tsk -> t_type = T_ATASK; /* indicate abreviated task control table */
  tsk -> t_wttk = 0;      /* indicate not waiting */
  tsk -> t_ratsk = sys_task; /* task family linkage */
  tsk -> t_pstsk = _tsk -> t_ldtsk;
  tsk -> t_nstsk =
  tsk -> t_fdtsk =
  tsk -> t_ldtsk = _NULL;
   _tsk -> t_ldtsk = tsk;
  If( tsk -> t_pstsk == _NULL ) tsk -> t_fdtsk = tsk;
  else tsk -> t_pstsk -> t_nstsk = tsk;
  if( prty > _tschd -> t_sclsl ) /* adjust priority */
    prty = _tschd -> t_sclsl-1;
  tsk -> t_cls = prty;    /* set priority */
  pq = _tschd -> t_sclsp[ prty ]; /* set scheduling array pointer */
  ++pq -> t_sct;          /* scheduling linkage */
  tsk -> t_pqtsk = pq -> t_lqtsk;
  tsk -> t_nqtsk = _NULL;
  if( tsk -> t_pqtsk == _NULL )pq -> t_fqtsk = tsk;
  else tsk -> t_pqtsk -> t_nqtsk = tsk;
  pq -> t_lqtsk = tsk;
  tsk -> t_xtm0 =         /* no execution time yet */
  tsk -> t_xtm1 =
  tsk -> t_xtm2 = 0;
  pr = sptr = ss+ssiz-2*pcnt; /* initialize execution stack & t_ps */
  tsk -> t_ps.sys_sp =
  sp = sptr - sizeof( struct t_xstck );
  sp -> t_xbp = ss + ssiz;
  sp -> t_xbase = ss;
  while( pcnt-- )*pr++ = *padr++;
  for( i = 0, s = &tsk -> t_ps; i++ < sizeof( struct sys_parm ); )*s = 0;
  tsk -> t_ps.sys_ds =
  tsk -> t_ps.sys_es =
  tsk -> t_ps.sys_ss =
  sp -> t_xds =
  sp -> t_xes =  _tsk -> t_ps.sys_ds;
  sp -> t_xdi =
  sp -> t_xsi =
  sp -> t_xdx =
  sp -> t_xcx =
  sp -> t_xbx =
  sp -> t_xax = _NULL;
  sp -> t_xip = xadr;
  tsk -> t_ps.sys_cs =
  sp -> t_xcs =
  _tsk -> t_ps.sys_cs;
  tsk -> t_ps.sys_pf =
  sp -> t_xpf = SYS_IF;
  tsk -> t_ps.sys_ip =
  sp -> t_retip = &t__halt;
  t__syxit( &crtflg );    /* exit system state */
  return tsk;             /* return */
err3: mm_free( tsk -> t_ps.sys_ss );
err2: mm_free( tsk );
err1: t__syxit( &crtflg );
  return _NULL;
}

/*
t__halt                     terminate task
*/
t__halt()
/*
If  a subtask returns into its orginal stack,  control will pass  to  t__halt.
This  function  deletes the subtask and then clears the sys_task pointer  just
before returning on the system stack (to reenter the security kernel). */
{
  logical haltflg;        /* system state flag storage */
  t__syntr( &haltflg );  /* enter system state */
  t_rtmark( &_tsk -> t_ratsk -> t_xtm0 ); /* accumulate pseudo time */
  t_del( sys_task, _F ); /* delete current task */
  sys_task = _NULL;       /* indicate task terminated */
  while(_T)t_term();      /* return to security kernel */
}

/*
t_del                       delete task
*/
t_del( tsk, st )
struct t_task *tsk;
int st;
/*
Task tsk is killed. st is the status returned to the calling program.
*/
{
#define tskf ( ( struct t_ftask * )tsk ) /** (t_ftask) */
  struct t_task *t;       /* task control table pointer */
  logical delflg;         /* system state flag storage */
  t__syntr( &delflg );    /* enter system state */
  while( ( t = tsk -> t_fdtsk ) )t_del( t, st ); /* delete subtasks first */
  if( tsk -> t_pstsk )    /* family linkage */
    tsk -> t_pstsk -> t_nstsk = tsk -> t_nstsk;
  else tsk -> t_ratsk -> t_ldtsk = tsk -> t_nstsk;
  if( tsk -> t_nstsk )
    tsk -> t_nstsk -> t_pstsk = tsk -> t_pstsk;
  else tsk -> t_ratsk -> t_fdtsk = tsk -> t_pstsk;
  if( tsk -> t_pqtsk )    /* queue linkage */
    tsk -> t_pqtsk -> t_nqtsk = tsk -> t_nqtsk;
  else if( ( tsk -> t_type&T_P )

       && ( _tschd -> t_sclsp[ tsk -> t_cls ] -> t_fqtsk == tsk ) )
          _tschd -> t_sclsp[ tsk -> t_cls ] -> t_fqtsk = tsk -> t_nqtsk;
  else if( ( tsk -> t_type&T_W ) && ( _tschd -> t_fwtsk == tsk ) )
    _tschd -> t_fwtsk = tsk -> t_nqtsk;
  else if( ( tsk -> t_type&T_SW ) && ( _tschd -> t_fswtsk == tsk ) )
    _tschd -> t_fswtsk = tsk -> t_nqtsk;
  else if( ( tsk -> t_type&T_X ) && ( _tschd -> t_fxtsk == tsk ) )
    _tschd -> t_fxtsk = tsk -> t_nqtsk;
```


(continued on page 58)

ISN'T IT A PITY...



Everything Isn't As Accommodating As

c-tree™ / r-tree™

FILE HANDLER | REPORT GENERATOR

Performance and Portability

For all the time you devote to developing your new programs, doesn't it make sense to insure they perform like lightning and can be ported with ease?

c-tree: Multi-Key ISAM Functions For Single User, Network, & Multi Tasking Systems

Based on the most advanced B+ Tree routines available today, **c-tree** gives you unmatched keyed file accessing performance and complete C Source Code. Thousands of professional C programmers are already enjoying **c-tree**'s royalty-free benefits, outstanding performance, and unparalleled portability.

Only FairCom provides single and multi-user capabilities in one source code package, including locking routines for Unix, Xenix, and DOS 3.1., for one low price! In addition, **c-tree** supports fixed and variable record length data files; fixed and variable length key values with key compression; multiple indices in a single index file; and automatic sharing of file descriptors.

r-tree: Multi-File Report Generator

r-tree builds on the power of **c-tree** to provide sophisticated, multi-line reports. Information spanning multiple files may be used for display purposes or to direct record selection. You can develop new reports or change existing reports without programming or recompiling and can use any text editor to create or modify **r-tree** report scripts including the complete report layout. At your option, end users may even modify the report scripts you provide.

Unlimited Virtual Fields; Automatic File Traversal

r-tree report scripts can define any number of virtual fields based on complex computational expressions involving application defined data objects and other virtual fields. In addition, **r-tree** automatically computes values based on the MAX, MIN, SUM, FRQ, or AVG of values spread over multiple records. **r-tree** even lets you nest these computational functions, causing files from different logical levels to be automatically traversed.

Unlike other report generators, **r-tree** allows you to distribute executable code capable of producing new reports or changing existing reports without royalty payments, provided the code is tied to an application. Your complete source code also includes the report script interpreter and compiler.

How To Order

Put FairCom leadership in programmers utilities to work for you. Order **c-tree** today for $395 or **r-tree** for $295. (When ordered together, **r-tree** is only $255). For VISA, MasterCard and C.O.D. orders, call 314/445-6833. For **c-tree** benchmark comparisons, write FairCom, 2606 Johnson Drive, Columbia, MO 65203.

Complete C Source Code & No Royalties!

Xenix is a registered trademark of Microsoft Corp. Unix is a registered trademark of AT&T.

Circle **no. 93** on reader service card.

DAN BRICKLIN'S DEMO PROGRAM

Read what they're saying about this new concept in prototyping and demo-making:

"A winner right out of the starting gate. After you use DEMO once, you'll wonder how you got along without it."
— PC Magazine, 4/29/86

"Everybody who writes software, either commercially or for in-house applications, should immediately order a copy. Period. No exceptions."
— Soft•letter, 4/20/86

"Its low price, superb performance, and range of applications practically guarantee that it will be widely used. Four Floppy Rating (8.0)"
— InfoWorld, 3/31/86

"Apparently has a hit on its hands with . . . a development tool for personal computer software that has won rave reviews from early users."
— Computerworld, 4/7/86

"A gem."
— PC Week, 3/18/86

Product of the Month
— PC Tech Journal, 3/86

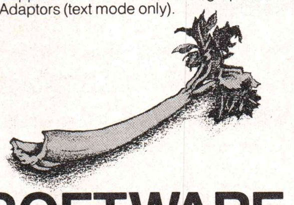
ORDER NOW!

Thousands of developers are designing better products faster and producing more effective demonstrations using Dan Bricklin's Demo Program. You can, too. Act now!

MasterCard **ONLY $74.95**
VISA **617-332-2240**

Massachusetts residents add $3.75. Outside of the U.S.A. add $15.00.

Requires 256k IBM PC/compatible, DOS 2.0 or later. Supports Monochrome, Color/graphics, and EGA Adaptors (text mode only).



SOFTWARE GARDEN

Dept. D
P.O. Box 373, Newton Highlands, MA 02161

Circle **no. 314** on reader service card.

MULTITASKING KERNEL

Listing One (Listing continued, text begins on page 16.)

```
  if( tsk -> t_nqtsk )tsk -> t_nqtsk -> t_pqtsk = tsk -> t_pqtsk;
  else if( ( tsk -> t_type&T_P )
       && ( _tschd -> t_sclsp[ tsk -> t_cls ] -> t_lqtsk == tsk ) )
         _tschd -> t_sclsp[tsk -> t_cls] -> t_lqtsk = tsk -> t_pqtsk;
  else if( ( tsk -> t_type&T_W ) && ( _tschd -> t_fwtsk == tsk ) )
    _tschd -> t_lwtsk = tsk -> t_pqtsk;
  else if( ( tsk -> t_type&T_SW ) && ( _tschd -> t_fswtsk == tsk ) )
    _tschd -> t_lswtsk = tsk -> t_pqtsk;
  else if( ( tsk -> t_type&T_X ) && ( _tschd -> t_fxtsk == tsk ) )
   _tschd -> t_lxtsk = tsk -> t_pqtsk;
  t = sys_task;           /* save current t_task pointer */
  sys_task = tsk -> t_ratsk; /* set ancestor task */
  mm_free( tsk -> t_ps.sys_ss ); /* free stack */
  mm_free( tsk );          /* free t_task table */
  sys_task = t;            /* restore current task pointer */
  t__syxit( &delflg );     /* exit system state */
  return _F;               /* return */
}

/*
mm_aloc                      memory allocation
*/
mm_aloc(ln)
unsigned ln;
/*
ln bytes are allocated from the heap. The address of the first byte is
returned. If there is not enough available memory to satisfy the request,
_NULL is returned. */
{
  return malloc(ln);       /* allocate storage */
}

/*
mm_free                      memory deallocation
*/
mm_free(st)
char *st;
/*
st is the address returned by a previous call to function mm_free. The storage
previously allocated is made available for future use. The normal return is
_F. _E is returned if st does not point to an area previously allocated by
mm_aloc. */
{
  return free(st);         /* deallocate storage */
}

/*
main                         test program
*/
main()
/*
This function serves to test the task scheduler. Two tasks are created, each
of which increments a variable. The original task continually displays the
counts, as well as its own iteration number. Depressing any key will cause a
return to MS-DOS. */
{
  int ctr1, ctr2, ctr3;  /* counters */
  int count();           /* counting subroutine */
  int param[ 2 ];        /* parameter array */
  printf("tasktest (C) 1986 Ken Berry- All Rights Reserved\n");
  printf("Tele task scheduler: 1986 September 2 version (DDJ mod)\n\n");
  t__init();             /* initialize task scheduler */
  ctr1 = ctr2 = ctr3 = 0; /* initialize counters */
  param[ 0 ] = &ctr1;    /* create first task */
  param[ 1 ] = 1;
  t_crt( count, 2, &param, 256, 0, 0, 0);
  param[ 0 ] = &ctr2;    /* create second task */
  param[ 1 ] = 2;
  t_crt( count, 2, &param, 256, 0, 0, 0);
  while( !kbhit() )      /* loop until key depressed */
  {
    ++ctr3;              /* increment main loop count */
    printf("main = %d, task 1 = %d, task 2 = %d\n", ctr3, ctr1, ctr2 );
  }
  getch();               /* discard termination character */
  t__term();             /* terminate task scheduler */
  return _F;             /* return to MS^DOS */
}

count(ctr,inc)
int *ctr,inc;
{
  while(_T)              /* infinite loop */
  {
    *ctr += inc;         /* update counter */
  }
}
```

End Listing One

Listing Two

```
Listing 2-     System Definitions
(C) Copyright 1986 Ken Berry.
All rights reserved.
Copies may be made for non-commercial, private use only.

sys_parm   struc          ;; register storage block
rax        dw ?           ;; ax (general register A)
rbx        dw ?           ;; bx (general register B)
rcx        dw ?           ;; cx (general register C)
rdx        dw ?           ;; dx (general register D)
rbp        dw ?           ;; bp (base pointer)
rsi        dw ?           ;; si (source index)
rdi        dw ?           ;; di (destination index)
```

```
rsp        dw ?              ;; sp (stack pointer)
rcs        dw ?              ;; cs (code segment)
rds        dw ?              ;; ds (data segment)
rss        dw ?              ;; ss (stack segment)
res        dw ?              ;; es (extra segment)
rpf        dw ?              ;; pf (processor flags)
rsw        dw ?              ;; sw (status word)
rip        dw ?              ;; ip (instruction pointer)
rres       dw ?              ;; unused
sys_parm   ends

t_task     struc             ;; task control table
t_type     db ?              ;; task type
t_wttk     dw ?              ;; wait tick count
t_cls      dw ?              ;; priority queue index
t_pqtsk    dw ?              ;; prior t_task pointer
t_nqtsk    dw ?              ;; next t_task pointer
t_ratsk    dw ?              ;; ancestor t_task pointer
t_pstsk    dw ?              ;; prior sibling t_task pointer
t_nstsk    dw ?              ;; next sibling t_task pointer
t_fdtsk    dw ?              ;; first desendant t_task pointer
t_ldtsk    dw ?              ;; last desendant t_task pointer
t_ps       db type sys_parm dup (?) ;; processor status
t_xtm0     dw ?              ;; *  execution time accumulator
t_xtm1     dw ?              ;; *
t_xtm2     dw ?              ;; *
t_axstk    dw ?              ;; application stack pointer
t_task     ends

T_ATSK     equ 01h           ;; abreviated task
T_X        equ 80h           ;; execute wueue
T_W        equ 40h           ;; wait queue
T_P        equ 20h           ;; priority queue
t_SW       equ 10h           ;; secondary wait queue

t_scls     struc             ;; scheduling class queue
t_sfrq     dw ?              ;; scheduling frequency
t_sct      dw ?              ;; queue length
t_fqtsk    dw ?              ;; first task in queue
t_lqtsk    dw ?              ;; last task in queue
t_scls     ends

t_schd     struc             ;; scheduling control table
t_xct      dw ?              ;; execution queue length
t_fxtsk    dw ?              ;; first task in execution queue
t_lxtsk    dw ?              ;; last task in execution queue
t_wct      dw ?              ;; wait queue length
t_fwtsk    dw ?              ;; first task in wait queue
t_lwtsk    dw ?              ;; last task in wait queue
t_swct     dw ?              ;; secondary wait queue length
t_fswtsk   dw ?              ;; first task in secondary wait queue
t_lswtsk   dw ?              ;; last task in secondary wait queue
t_sclsl    dw ?              ;; scheduling class index limit
t_sclsp    dw ?              ;; scheduling class array pointer
t_schd     ends

t_calln    struc             ;; near function call
t_nbp      dw ?              ;; base pointer storage
t_nret     dw ?              ;; return address
t_np0      dw ?              ;; parameter 0
t_np1      dw ?              ;; parameter 1
t_np2      dw ?              ;; parameter 2
t_np3      dw ?              ;; parameter 3
t_np4      dw ?              ;; parameter 4
t_np5      dw ?              ;; parameter 5
t_np6      dw ?              ;; parameter 6
t_np7      dw ?              ;; parameter 7
t_calln    ends

t_xtsk     struc             ;; execution stack
t_xbase    dw ?              ;; _base (for stack overflow detection)
t_xes      dw ?              ;; es
t_xbp      dw ?              ;; bp
t_xdi      dw ?              ;; di
t_xsi      dw ?              ;; si
t_xdx      dw ?              ;; dx
t_xcx      dw ?              ;; cx
t_xbx      dw ?              ;; bx
t_xax      dw ?              ;; ax
t_xds      dw ?              ;; ds
t_xip      dw ?              ;; ip
t_xcs      dw ?              ;; cs
t_xpf      dw ?              ;; pf
t_retip    dw ?              ;; return ip
t_xtsk     ends

retn       macro s           ;; near return
  ifnb <s>
           db 0C2h           ;; pop ip & adjust sp
           db high s         ;; *  adjustment value
           db low s          ;; *
  else
           db 0C3h           ;; pop ip only
  endif
           endm

retf       macro s           ;; far return
  ifnb <s>
           db 0CCh           ;; pop ip, cs & adjust sp
           db high s         ;; *  adjustment value
           db low s          ;; *
  else
           db 0CBh           ;; pop ip, cs only
  endif
           endm

ilck       macro reg,flag
           xchg reg,flag  ;; capture token
           endm

iowait     macro
           nop               ;; I/O delay
           endm
```

(continued on next page)

Listing Two **(Listing continued, text begins on page 16.)**

```
sys_entr  macro flag      ;; enter system function
  ifndef sys_ilck
          extrn sys_ilck:byte
  endif
          mov al,0FFh     ;; ** system task interlock
          ilck al,sys_ilck ;; **
          mov flag,al     ;; save asynchronous status
          endm

sys_exit  macro flag      ;; exit from system function
          local exit1,exit2
  ifndef sys_ilck
          extrn sys_ilck:byte
  endif
  ifndef t_astrm
          extrn t_astrm:byte
  endif
  ifndef t_term
          extrn t_term:near
  endif
          test byte ptr t_astrm,0FFh ;; *  test for application terminated
          jnz exit1       ;; *
          mov byte ptr sys_ilck,0 ;; exit system state
          jmp short exit2 ;; continue application task
exit1:    test flag,0FFH ;; ** test for more stacked system tasks
          jnz exit2       ;; **
          call t_term     ;; terminate application task
exit2:                    ;; macro exit
          endm

sys_sync  macro flag      ;; synchronize system resource
  ifndef t_sync
          extrn t_sync:near
  endif
          lea bx,flag     ;; set flag offset
          call t_sync     ;; suspend task until token obtained
          endm

sys_sstk  macro           ;; conditionally establish system stack
          local sstk1
  ifndef t__sstk
          extrn t__sstk:near
  endif
          or al,al        ;; *  test for system task interrupted
          jnz sstk1       ;; *
          call t__sstk    ;; establish system stack
sstk1:    push ds         ;; ** set es = ds
          pop es          ;; **
          endm

sys_sctx  macro           ;; save processor context
          push bx         ;; protect bx
          push cx         ;; protect cx
          push dx         ;; protect dx
          push si         ;; protect si
          push di         ;; protect di
          push bp         ;; protect bp
          push es         ;; protect es
          cld             ;; clear direction flag
          sys_sstk        ;; conditionally establish system stack
          endm

sys__rctx macro           ;; restore processor context (except ds)
          pop es          ;; restore es
          pop bp          ;; restore bp
          pop di          ;; restore di
          pop si          ;; restore si
          pop dx          ;; restore dx
          pop cx          ;; restore cx
          pop bx          ;; restore bx
          pop ax          ;; restore ax
          endm

sys_rctx  macro           ;; restore processor context
          sys__rctx       ;; restore context (except ds)
          pop ds          ;; restore ds
          endm

sys_ient  macro flag
          push ds         ;; protect ds
          push ax         ;; protect ax
          mov ax,dgroup   ;; *  establish data addressability
          mov ds,ax       ;; *
          sys_entr flag   ;; enter system state
          sti             ;; interrupts on
          sys_sctx        ;; save processor context
          endm

sys_iret  macro flag
          local iret1
  ifndef t_astrm
          extrn t_astrm:byte
  endif
          cli             ;; interrupts off
          test byte ptr t_astrm,0FFh ;; *  test for application not terminated
          jz iret1        ;; *
          test flag,0FFh ;; ** test for system state interrupted
          jnz iret1       ;; **

          sti             ;; interrupts on
          retn            ;; return to task management
iret1:    sys_rctx        ;; restore processor context
          iret            ;; resume interrupted task
          endm

dseg      macro
dgroup    group data
data      segment word public 'data'
```

```
          assume ds:dgroup,es:dgroup,ss:dgroup
          endm

endds     macro
data      ends
          endm

pseg      macro
pgroup    group prog
prog      segment byte public 'prog'
          assume cs:pgroup
          endm

endps     macro
prog      ends
          endm
```

End Listing Two

Listing Three

```
Listing 3      Scheduling Algorithm (Assembly Subroutines)
(C) Copyright 1986 Ken Berry.
All rights reserved.
Copies may be made for non-commercial, private use only.

          include tele.mac ; system definitions (listing 2)

          extrn t_astrm:byte ; application task termination flag
          extrn t_rtmark:near ; update pseudo time accumulator
          extrn t__krnl:near ; security kernel

          public sys_task ; current task pointer
          public sys_tsch ; task scheduling table pointer
          public sys_ilck ; system task interlock
          public sys_asbs ; application stack base
          public sys_dgrp ; data segment storage
          public sys_ssbs ; system stack base
          public sys_sssz ; system stack size
          public sys_sstp ; system stack top
          public sys_stat ; original register block
          public t_mnxtm ; minimum execution time
          public t_rels  ; release
          public t_spnd  ; suspend
          public t_syxtm ; system pseudo time accumulator
          public t_term  ; reschedule
          public t_wait  ; wait
          public t_wqupd ; wait queue update task pointer
          public t__crtss ; current task ss storage
          public t__crtsp ; current task sp storage
          public t__dspap ; dispatch application
          public t__dspsy ; dispatch system
          public t__sstk ; establish system stack
          public t__syntr ; enter system state
          public t__syxit ; exit system state

MINXTM    equ 500          ; minimum execution time
STKLN     equ 1024         ; system stack size

          dseg

tmrdx     dw 0             ; dx storage
spdss     dw 0             ; ss storage
spdsp     dw 0             ; sp storage

t__crtss  dw 0             ; current task ss storage
t__crtsp  dw 0             ; current task sp storage

sys_stat  db type sys_parm dup (0) ; original register block
sys_dgrp  dw 0             ; data segment storage
sys_task  dw 0             ; current task pointer
sys_tsch  dw 0             ; task scheduling table pointer
sys_asbs  dw 0             ; application stack base
sys_ssbs  dw stkbs         ; system stack base
sys_sssz  dw STKLN         ; system stack length
sys_sstp  dw STKLN         ; system stack top
sys_ilck  db 0FFh          ; system task interlock

t_wqupd   dw 0             ; wait queue update task pointer

t_syxtm   dw 3 dup (0)     ; system pseudo time accumulator

t_mnxtm   dw MINXTM        ; minimum execution time

stkbs     db STKLN dup (0) ; system stack
          db type t_task dup (0) ; main task control table

          endds

          pseg

comment ~

t__dspap(ss,sp)
selector ss;
unsigned sp;

ss  and  sp are placed in the stack registers.  Then the other  registers  are
restored from the new stack.  Control passes to the restored task.  The return
address  is left at the top of the system stack.  Therefore the restored  task
may use the system stack to return to the caller of t__dspap. ax may contain a
return code in this case.
~

t__dspap  proc near
          push bp          ; protect bp
          mov bp,sp        ; establish parameter addressability
          mov ax,[bp].t_np0 ; set application stack
          mov bx,[bp].t_np1
```

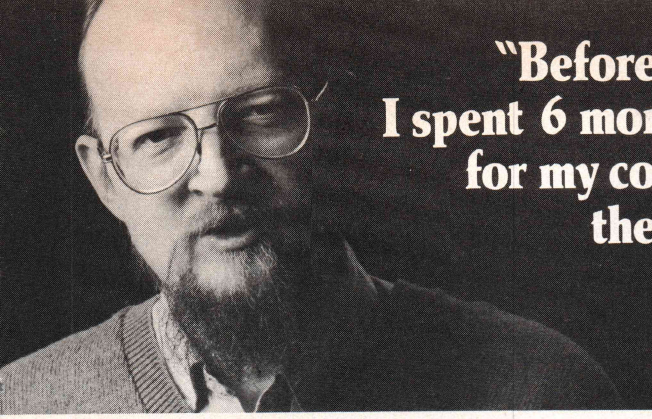
(continued on next page)

MULTITASKING KERNEL

Listing Three (Listing continued, text begins on page 16.)

```
        mov sys_sstp,sp ; store current top of system stack
        cli
        mov ss,ax
        mov sp,bx
        mov bp,sp       ; enable interrupts
        or [bp].t_xpf,0200h
        pop sys_asbs
        sti
        sys__rctx       ; restore context
        cli             ; interrupts off
        mov byte ptr sys_ilck,0 ; exit system state
        mov byte ptr t_astrm,0 ; initialize application interval
        pop ds          ; restore ds
        iret            ; execute task
t__dspap  endp

comment ~

t_term()        _F
t_spnd(tp)      tp
t_wait(tp)      tp
unsigned tp;
t_rels()        _E

All of these functions are similar.  The processor registers are stored on
the stack,  which  is  then  adjusted  to  match the pattern for interrupt
returns. Finally the system stack is established.  The  functions  differ  in
the  code returned  to  the caller of function t__dspap.  t__dspap restores
the registers and returns control to the caller of these functions.  The
returned  value  is shown with the appropriate call above.  tp is only used
with t_spnd and t_wait.  It is the number of system ticks to  wait  before
executing  the  task  again.  t_wait functions like t_spnd, except that t_rels
is invoked immediately.  ~

t_term    proc near
          call t__trmap  ; protect registers
          xor ax,ax      ; return _F
          ret
t_term    endp

t_spnd    proc near
          mov spdss,ss   ; store stack pointers
          mov spdsp,sp
          call t__trmap  ; protect registers
          mov es,spdss   ; return tick count
          mov si,spdsp
          mov ax,word ptr es:[si+2]
          push ds        ; set es = ds
          pop es
          ret            ; return
t_spnd    endp

t_wait    proc near
          push bp        ; protect bp
          mov bp,sp      ; establish stack addressability
          mov ax,[bp].t_np0 ; suspend task
          push ax
          call t_spnd
          mov sp,bp      ; unload stack
          pop bp         ; restore bp
t_rels    proc near
          call t__trmap  ; protect registers
          xor ax,ax      ; return _E
          dec ax
          ret
t_rels    endp
t_wait    endp

comment ~

t__dspsy()

A call to function t__trmap is made so that after the registers are stored in
the  application stack (and the system stack is made current),  control passes
to  function t__krnl,  the system security kernel.  Control will  return  from
t__dspsy when the calling task is resumed. Nothing is returned.
~

t__dspsy  proc near
          mov ax,offset pgroup:t__krnl ; branch to system
          push ax
          sub sys_sstp,2 ; adjust system stack (for "pop bp" in t__sstk)

comment ~

t__trmap()

The  machine registers are stored on the application stack.  Then the system
stack is made current.  The return address from the call to t__trmap is put on
the system stack before returning to it. Nothing is returned.
~

t__trmap  proc near
          mov byte ptr sys_ilck,0FFh ; force system state
          mov tmrdx,dx    ; save dx
          pop dx          ; set return address (from t__trmap)
          push cs         ; protect cs
          pushf           ; protect flags
          push ds         ; protect ds
          push ax         ; protect ax
          push bx         ; protect bx
          push cx         ; protect cx
          push tmrdx      ; protect dx
          push si         ; protect si
          push di         ; protect di
          push bp         ; protect bp
```

(continued on page 64)



"Before I chose Microsoft C, I spent 6 months evaluating C compilers for my company. Now you can do the same in 2 hours."

Bill Davidsen
Software Engineer
Real Time and Operating Systems

"Call us. You can get Microsoft C or our comprehensive report on C by the day after tomorrow."

Bruce Lynch, President
The Programmer's Shop

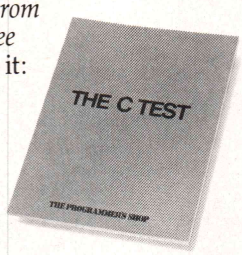
The security of thorough research. It took Bill Davidsen six months to thoroughly evaluate all C products before he selected Microsoft C. For him, its tight code and UNIX System V™ compatibility were exactly what he needed. And now Version 4.00 includes CodeView,™ a source-level windowing debugger.

Thanks to expert users like Bill, and The Programmer's Shop, you can enjoy that satisfied feeling of thorough product evaluation in just a few hours.

We recommend evaluating software by also getting detailed information from several different sources, including unbiased reports and reviews. Bill agrees completely.

In fact, he helped us compile the objective opinions of 4 magazines, 14 users and 3 industry analysts in a 16-page report on C: *The C Test.* It can help you be absolutely sure of making the choice that's best for you. And it's absolutely free.

C for yourself. As an objective evaluation by users and professionals alike, *The C Test* is one of the most comprehensive and informative reports currently available on C development tools. *It's only available from The Programmer's Shop. And it's yours free for the asking.* Here's what you'll find in it:



The C Test ▪ Detailed Tech Specs ▪ Benchmark Source Code ▪ Magazine Reviews ▪ Users' Feedback ▪ Performance Benchmarks ▪ User Study and Profiles ▪ Test Drive Survey Results ▪ 37 Compatible Products

And if you're looking for even more C support, Microsoft-compatible libraries for file management, graphics, screen control, object-oriented programming and other tools are ready to ship.

The best programs for less. We think the only way to serve you is to give you the best programming alternatives. The best recommendations for your needs. To deliver immediately. And this is how we do it.

We start by giving you a choice of over 62 programming language implementations and 174 support programs. All from the same source. All competitively priced.

Our informed programmers offer free advice whenever you call with any questions about any product.

And when you place an order, we can rush it to you in 48 hours or less. That's the kind of service and support our 10,000 customers have come to expect.

Because we've become a success by giving the best advice for free and selling the best software for less.

To order Microsoft C ($279) or for your free copy of *The C Test*, simply call the toll-free number below:

1-800-421-8006. In Massachusetts, call 1-800-442-8070.

MICROSOFT® C Compiler Version 4.00

MICROSOFT C COMPILER
- Produces fast executables and optimized code including elimination of common sub-expressions. NEW!
- Implements register variables.
- Small, Medium and Large Memory model libraries.
- Compact and HUGE memory model libraries. NEW!
- Can mix models with NEAR, FAR and the new HUGE pointers.
- Library routines implement most of UNIX System V C library.
- Start-up source code to help create ROMable code. NEW!
- Full proposed ANSI C library support (except clock). NEW!
- Link your C routines with Microsoft FORTRAN (version 3.3 or higher), Microsoft Pascal (version 3.3 or higher) or Microsoft Macro Assembler.
- Microsoft Windows support and MS-DOS 3.1 networking support.

MICROSOFT PROGRAM MAINTENANCE UTILITY. NEW!
- Rebuilds your applications after your source files have changed.
- Supports macro definitions and inference rules.

OTHER UTILITIES.
- Library Manager.
- Overlay Linker.
- EXE File Compression Utility.
- EXE File Header Utility.

MICROSOFT CodeView WINDOW-ORIENTED SOURCE-LEVEL DEBUGGER. NEW!
- Watch the values of your local and global variables and expressions as you debug.
- Set conditional breakpoints on variables, expressions or memory; trace and single step.
- Watch CPU registers and flags as you execute.
- Debug using your original source code, the resulting disassembly or both intermingled.

Microsoft C comes with a 30-day money-back guarantee from The Programmer's Shop.

THE PROGRAMMER'S SHOP

The programmer's complete source for software, services and answers.

128 Rockland Street, Hanover, MA 02339 **(617) 826-7531**

Circle **no. 174** on reader service card.

MULTITASKING KERNEL

Listing Three **(Listing continued, text begins on page 16.)**

```
            push es           ; protect es
            push dx           ; restore return address to stack
            mov bp,sp         ; establish stack addressability
            mov ax,[bp].t_xip ; adjust stack for interrupt return
            xchg ax,[bp].t_xpf
            mov [bp].t_xip,ax

comment ~

t__sstk()

The  current application stack pointers are stored.  Then the system stack is
established as the current stack.  The return address from the call is placed
on the system stack before returning into it. Nothing is returned.
~

t__sstk     proc near
            pop dx            ; unload return address
            push sys_asbs     ; protect stack protection reference
            mov bx,ss         ; set application stack registers
            mov cx,sp
            mov ax,sys_ssbs ; set system stack
            cli
            mov sys_asbs,ax
            push ds
            pop ss
            mov sp,sys_sstp
            sti
            pop bp            ; restore bp
            mov t__crtss,bx ; store current ss
            mov t__crtsp,cx ; store current bp
            push dx           ; return to caller
            ret
t__sstk     endp
t__trmap    endp
t__dspsy    endp

comment ~

t_sync(flg)
char *flg;

A wait loop will be entered until the required resource is available. This is
indicated  by flg containing 0x00.  0xFF is stored to prevent any other tasks
from acquiring the resource. The resource is released by resetting flg to
0x00.
~

t_sync      proc near
            push bp           ; protect bp
            mov bp,sp         ; establish stack addressability
            mov bx,[bp].t_np0 ; set pointer to resource flag

sync1:      mov al,0FFh       ; interlock token
            ilck al,<byte ptr [bx]>
            or al,al          ; test for token acquired
            jz sync2
            xor ax,ax         ; wait for 1 system tick
            inc ax
            push ax
            call t_spnd
            mov sp,bp
            jmp sync1         ; continue

sync2:      pop bp            ; restore bp
            call t_rels       ; release task
            ret               ; return
t_sync      endp

comment ~

t__syntr(flg)
char *flg;

This function expands the sys_entr macro for use by c functions.
~

t__syntr    proc near
            push bp           ; protect bp
            mov bp,sp         ; establish stack addressability
            mov bx,[bp].t_np0 ; set flag address
            sys_entr <byte ptr [bx]> ; enter system state
            pop bp            ; restore bp
            ret               ; return
t__syntr    endp

comment ~

t__syxit(flg)
char *flg;

This function expands the sys_exit macro for use by c functions.
~

t__syxit    proc near
            push bp           ; protect bp
            mov bp,sp         ; establish stack addressability
            mov bx,[bp].t_np0 ; set flag address
            sys_exit <byte ptr [bx]> ; exit system state
            pop bp            ; restore bp

            ret               ; return
t__syxit    endp

            endps

            end
```

End Listing Three

(Listing Four begins on page 66)

A MEGABYTE FOR DOS!

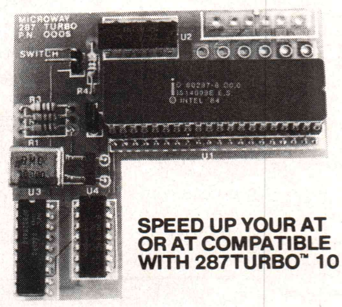
MicroWay is the world's leading retailer of 8087s and high performance PC upgrades. We stock a complete selection of 8087s that run from 5 to 12 MHz. All of our coprocessors are shipped with a diagnostic disk and the best warranty in the business – one year! We also offer daughterboards for socketless computers (NEC PC) and 287Turbo which increases the clock speed of the 80287 from 4 to 10 MHz. Our NUMBER SMASHER/ECM™ runs at 12 MHz with a megabyte of RAM and achieves a throughput of .1 megaflops with 87BASIC/INLINE, Intel Fortran, or Microsoft Fortran. Software reviewers consistently cite MicroWay software and 8087 expertise as the best in the industry! Our customers frequently write to thank us for recommending the correct software and hardware to meet their specific needs. They also thank us for our same-day shipping! In addition to our own products which support the 8087 and 80287, we stock the largest supply of specialized software available. For more information call us at

617-746-7341

NUMBER SMASHER/ECM™ THE FASTEST ACCELERATOR CARD AVAILABLE

gives you 12 MHz speed in two modes: 704K or one megabyte of "Extended Conventional Memory." MEGASWITCH MMU and MegaDOS software make it possible to run DOS applications with up to 1015K using PC compilers, AutoCAD and Lotus 1-2-3. Does not require EMS software. Totally compatible ... from **$599** for 512K and 9.54 MHz to **$1199** with one megabyte and 8087-12.

Cooling Fan (for 12 MHz)........ **$80**



SPEED UP YOUR AT OR AT COMPATIBLE WITH 287TURBO™ 10 MHz

Micro Way® 8087 Support

For the IBM PC, PC XT, PC AT and Compatibles.

A2D-160™ MicroWay's Data Acquisition Board performs 160,000 12 bit Analog to Digital conversions per second! Includes software drivers. The fastest 12 bit A to D board available. For the IBM PC XT and compatibles.................. **$1295**

87SFL™ Accurate and fast **Scientific Subroutine Packages.** More than 145 functions: 34 Elementary Fcns, 18 Probability and Statistical Fcns, 15 Random Number Generators, $Ei(x)$, $E_n(x)$, $li(x)$, $Si(x)$, $Ci(x)$, $\Gamma(x)$, $\psi(x)$, $B(x,\omega)$, $I_x(a,b)$, erf x, $S(x)$, $C(x)$, $J_\nu(z)$, $Y_\nu(x)$, $I_\nu(z)$, $K_\nu(x)$, $Ai(x)$, $Bi(x)$, $Ai'(x)$, $Bi'(x)$, ber x, bei x, $\ker_1 x$, kei′ x, $K(x)$, $E(x)$, $F(\rho|a)$, $E(\rho|a)$, $\Pi(\rho|a,b)$, $\Lambda(a,b,\rho)$, $\mathscr{P}(z)$, $\mathscr{P}'(z)$, $P_n(x)$, $H_n(x)$, $L_n^{(\alpha)}(x)$, $J_n^{(\alpha,\beta)}(x)$, $G_n(p,q,x)$, $C_n^{(\alpha)}(x)$ and many more. Callable from FORTRAN, Pascal, BASIC and C. 1st module $250, 2nd on $100.

MATRIXPAK™ manages a **MEGABYTE!** Written in assembly language, our runtime package accurately manipulates large matrices at very fast speeds. Includes matrix inversion and the solution of simultaneous linear equations. Callable from RM or MS Fortran, MS Assembler, or 87BASIC/INLINE..................... each **$99**

87FFT™ Written in assembly language, performs Forward and Inverse FFTs on real and complex arrays which occupy up to 512 Kbytes of RAM. Also does convolutions, auto correlations, hamming, complex vector multiplication, and complex to radial conversions. Callable from most 8087 compatible compilers............... **$200**

87FFT-2™ performs two-dimensional FFTs. Ideal for image processing. Requires 87FFT **$100**

FASTBREAK™ 8087 support for Lotus 1-2-3™ Version 1A or 1A*.................. **$79**

Microsoft Fortran V 3.31	**$209**
IBM Professional Fortran	**$565**
Ryan-McFarland Fortran V 2.0	**$399**
LAHEY Fortran F77L	**$477**
Grafmatic for Fortran or Pascal	**$125**
MultiHalo Graphics (1 language)	**$189**
Microsoft C V 4.0	**$299**
STSC APL★PLUS/PC	**$450**
INTEL, JRAM, AST, MAYNARD	**CALL**

Micro Way
P.O. Box 79
Kingston, Mass.
02364 USA
(617) 746-7341

MegaPage™ The only Intel-Lotus EMS board which comes with two megabytes of cool-running, low power drain CMOS RAM installed. Includes RAM disk, print spooler, disk cache, and EMS drivers. For the IBM PC, XT & compatibles.... **$549**

DFixer™ Our disk utility which thoroughly checks PC or AT hard disks for bad sectors and updates the MS DOS file allocation table accordingly. Solves the AT hard disk problem! ... **$149**

DOptimizer™ Optimizes the way your hard disk or floppy stores its files. Speeds up accesses by recombining fragmented files........... **$49**

DCache™ Our disk caching software speeds up your I/O by storing repetitively used tracks in memory. The amount of memory used can be selected in 64 Kbyte banks............... **$49**

87MACRO/DEBUG™ Contains all the pieces needed for writing 8087/80287 assembly code & MicroWay's 87DEBUG debugger. **$199**

OBJ→ASM™ A multipass object module translator and disassembler. Produces assembly language listings which include public symbols, external symbols and labels commented with cross references. Ideal for patching object modules for which source is not available............. **$200**

87BASIC™ includes patches to the IBM BASIC or MS Quick BASIC Compiler for USER TRANSPARENT 8087 support. Provides super fast performance for all numeric operations including trigonometrics, transcendentals, addition, subtraction, multiplication, and division...... each **$150**

87BASIC/INLINE™ converts the output of the IBM BASIC Compiler into optimized 8087 inline code which executes up to seven times faster than 87BASIC. Supports separately compiled inline subroutines. Requires the IBM BASIC Compiler Version 1 and a Macro Assembler. Includes 87BASIC **$200**

RTOS - REAL TIME OPERATING SYSTEM RTOS is a multi-user, multi-tasking real time operating system. It includes a configured version of Intel's iRMX-86, LINK-86, LOC-86, LIB-86, OH-86, and MicroWay's 87DEBUG. Runs on the IBM-PC, XT, PC-AT and COMPAQ.................. **$600**

INTEL COMPILERS[1]

FORTRAN-86	**$750**
PASCAL-86	**$750**
PL/M-86	**$500**
ASM-86	**$200**

[1]Requires RTOS or iRMX-86. All Intel compiler names and iRMX-86 TM Intel Corp.

8087 UPGRADES

All MicroWay 8087s include a one year warranty, complete MicroWay Test Program and accurate installation instructions.

8087 5 MHz............... **$114**
For the IBM PC, XT and compatibles.

8087-2 8 MHz........... **$149**
For Wang, AT&T, DeskPro, NEC, Leading Edge.

80287-3 5 MHz......... **$179**
For the IBM PC AT and 286 compatibles.

80287-6 6 MHz......... **$229**
For 8 MHz AT compatibles.

80287-8 8 MHz......... **$295**
For the 8 MHz 80286 accelerator cards.

NEC V20, V30	**$16, $30**
64K RAM Set 150ns	**$10**
256K RAM Set 150ns	**$29**
256K RAM Set 120ns	**$39**
128K RAM Set PC AT	**$49**

287Turbo™ 10 MHz If you own an AT, Deskpro 286 or AT compatible, this is the card you need to get reasonable numeric performance. It plugs into your 80287 socket and includes a specially driven 10 MHz 80287. The card comes in three configurations. The IBM AT version includes a hardware RESET button................ **$450**

287Turbo 8 MHz................... **$369**

87/88Turbo™ is a stubby card which includes a clock calendar and a speed controller which changes the speed of your motherboard from 4.77 to 7.4 MHz. Its use requires your PC to have a socketed 8284. Typical speed increase is 1.6 to 2.0. The card overcomes slow hardware by slowing up only when such devices are accessed and running at full speed otherwise **$149**
Optional 8087-2.......................... **$149**

286TurboCache™ This accelerator for the PC or XT uses 8K of cache memory and 80286/80287 processors to provide an average speed increase of 3:1 for most programs. Call for specifications and benchmarks................. **$469**

Call for our complete catalog of software which supports the 8087.

In London, please phone 223-7662

Circle **no. 300** on reader service card.

MULTITASKING KERNEL

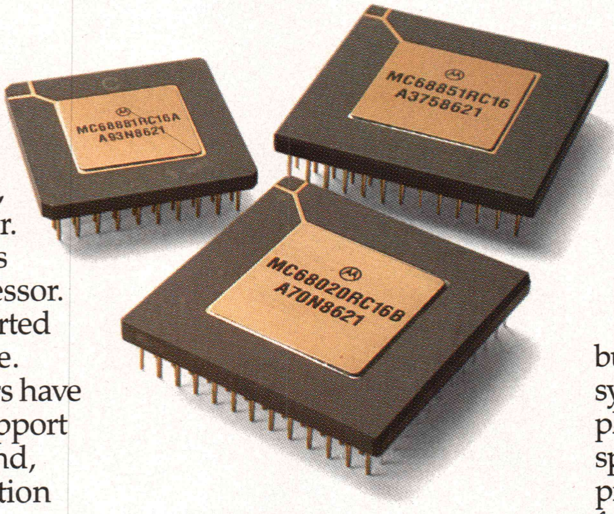
Listing Four (Listing continued, text begins on page 16.)

```
Listing 4       High Resolution Clock
(C) Copyright 1986 Ken Berry.
All rights reserved.
Copies may be made for non-commercial, private use only.
~

        include tele.mac ; system defintions (listing 2)

        extrn t_syxtm:word ; system execution time accumulator
        extrn sys_dgrp:word ; data segment storage
        extrn sys_stat:word ; original register block

        public t__tick ; system tick interrupt service
        public t_astrm ; application task termination flag
        public tmr_dspl ; physical display pointer
        public tmr_dvsr ; timer period
        public tmr_ilck ; tick service reentrant interlock
        public tmr_sync ; synchronization function address
        public tmr_tkct ; tick clock
        public tmr_xtm ; tick service execution time
        public tmr__clr ; reset time base generation
        public tmr__int ; timer initialization function
        public tmr__rst ; timer termination function
        public tmr__sts ; read timer status
        public tmr__tmr ; restart hardware timer
        public t_rdclk ; read high resolution clock
        public t_rtactg ; psuedo time accumulator pointer
        public t_rtmark ; mark execution interval
        public t__rdclk ; read real time clock
        public td_ref ; clock update tick reference count
        public td_tct ; clock tick timer
        public td__set ; set time of day clock
        public td__upd ; update time of day clock
        public w__cdspl ; physical display update function
        public w__sync ; physical display synchronization

RLCINT  equ 80h         ; relocated alternate time base interrupt
TMRINT  equ 8           ; hardware timer interrupt
TMRPRT  equ 40h         ; timer (8253) port
TMRPRD  equ 19912       ; timer period (60 Hz rate)
;TMRPRD  equ 9956        ; timer period (120 Hz rate)
INTPRT  equ 20h         ; interrupt controller (8259) port
TMRMSK  equ 01h         ; hardware timer interrupt mask
INTEOI  equ 20h         ; interrupt termination value
DSPCT   equ 1           ; 60 Hz interrupt rate
;DSPCT   equ 2           ; 120 Hz interrupt rate
IDV0    equ 3           ; tmr_idv0 divisor
ISKP0   equ 776         ; tmr_ict correction value
ISKP1   equ 11          ; tmr_idv1 correction value
ISKP2   equ 38          ; tmr_idv2 correction value

        dseg

tmr_tkct dw 0           ; interrupt counter
tmr_dct  db 0           ; display counter
tmr_ict  dw 0           ; tick clock (for time base generation)
tmr_dvsr dw TMRPRD      ; 1/2 timer period
t_astrm  db 0FFh        ; application task termination flag
tmrflg   db 0FFh        ; system state flag (t__tick)
tmr_ilck db 0           ; tick service reentrant interlock
tmr_idv0 db 0           ; clock time base generator
tmr_idv1 db 0           ; primary alternate time base generator
tmr_idv2 db 0           ; secondary alternate time base generator
tmr_dspl dw 0           ; console display w_pwdw pointer
t_rtactg dw 0           ; psuedo time accumulator pointer
t_rtrfct dw 0           ; real time reference count
t_rttick dw 0           ; tick clock phase
tmr_xtm  dw 3 dup (0)   ; tick service psuedo time accumulator
tmrpxtm  dw 0           ; prior psuedo time accumulator pointer
tmr_sync dw offset pgroup:w__sync ; synchronization function pointer
td_ref   dw 0           ; clock update tick reference count
td_tct   dw 0           ; clock tick timer

        endds

        pseg

comment ~
t__tick                         system tick service

t__tick\\

Control only comes here in response to an interrupt from the system clock.
This function serves three purposes. It maintains the system clock, which
provides the current date and time for both system and application uses. It
also  performs an update of the first physical display. And finally it
terminates the execution interval for the current application task.
~

t__tick  proc far

; reentrant lockout

        assume ss:nothing,ds:nothing,es:nothing
        sti             ; interrupts on
        push ds         ; protect ds
        push ax         ; protect ax
        mov ax,dgroup   ; establish data addressability
        mov ds,ax
        assume ds:dgroup
        mov al,INTEOI   ; terminate interrupt
        out INTPRT,al
        ilck al,tmr_ilck ; test for not reentrant call
        or al,al
        jz tick
        pop ax          ; restore ax
```

(continued on page 74)


SOFTWARE COMPONENTS Group
MOTOROLA
PDOS
GEM™/GEM™ DOS 68K
DIGITAL RESEARCH®
RM/COBOL™
Pascal 2
Oregon Software
RTUX™
Concurrent™ DOS
polyFORTH II
Technical Documentation
UNIX
SYSTEM V/68
MOTOROLA MICROSYSTEMS
SYSTEM V/68
Volume IV
VRTX/68000 USER'S GUIDE
PL/M
ELESOFT ADA
FORTRAN 77
FRANZ LISP
Reference Manual
Technical Documentation
VERSAdos
MOTOROLA MICROSYSTEMS
VERSAdos
Volume I
BASIC
Whitesmiths, Ltd.
C
MPROLOG
MODULA-2
Graphic Kernel System GK-2
MACRO ASSEMBLER 68000/68010/68020

Put your chips on the software.



The best bet.

If you're betting your future on a 32-bit microprocessor-based system, you'd rather bet a winner. Today, the clear winner is the MC68020 microprocessor. It's faster. It's fully supported in hardware and software. It's backed by what others have called, "The strongest support team in the business." And, we're delivering production quantities. Now.

In fact, last year Motorola shipped over 50,000 units and this year we expect to ship more than a quarter of a million MC68020s.

Software support. Now.

When you choose the MC68020, you get the largest body of software available for 32-bit microprocessor-based machines. Operating systems. Languages. Tools and utilities. Host independent development systems. And, of course, applications. In fact, 80% of the microprocessor-based system designs running UNIX® OSs are based on M68000 Family chips.

The fastest just got faster.

At 25 MHz, the latest version of the MC68020 and its earlier 20 MHz version remain

The MC68020

the fastest general-purpose processors on the market today. And fully compatible. But speed alone isn't the answer. And for the generation beyond the MC68020, we're committed to higher performance devices while maintaining the M68000 Family software base.

Chips, boards or boxes.

Motorola is with you at every level. As the world's leader in VMEbus products, supplying MC68020-based VMEmodule™ and VMEsystem™ products, we can work with your development team at virtually every level, including technical training.

Betting the winner.

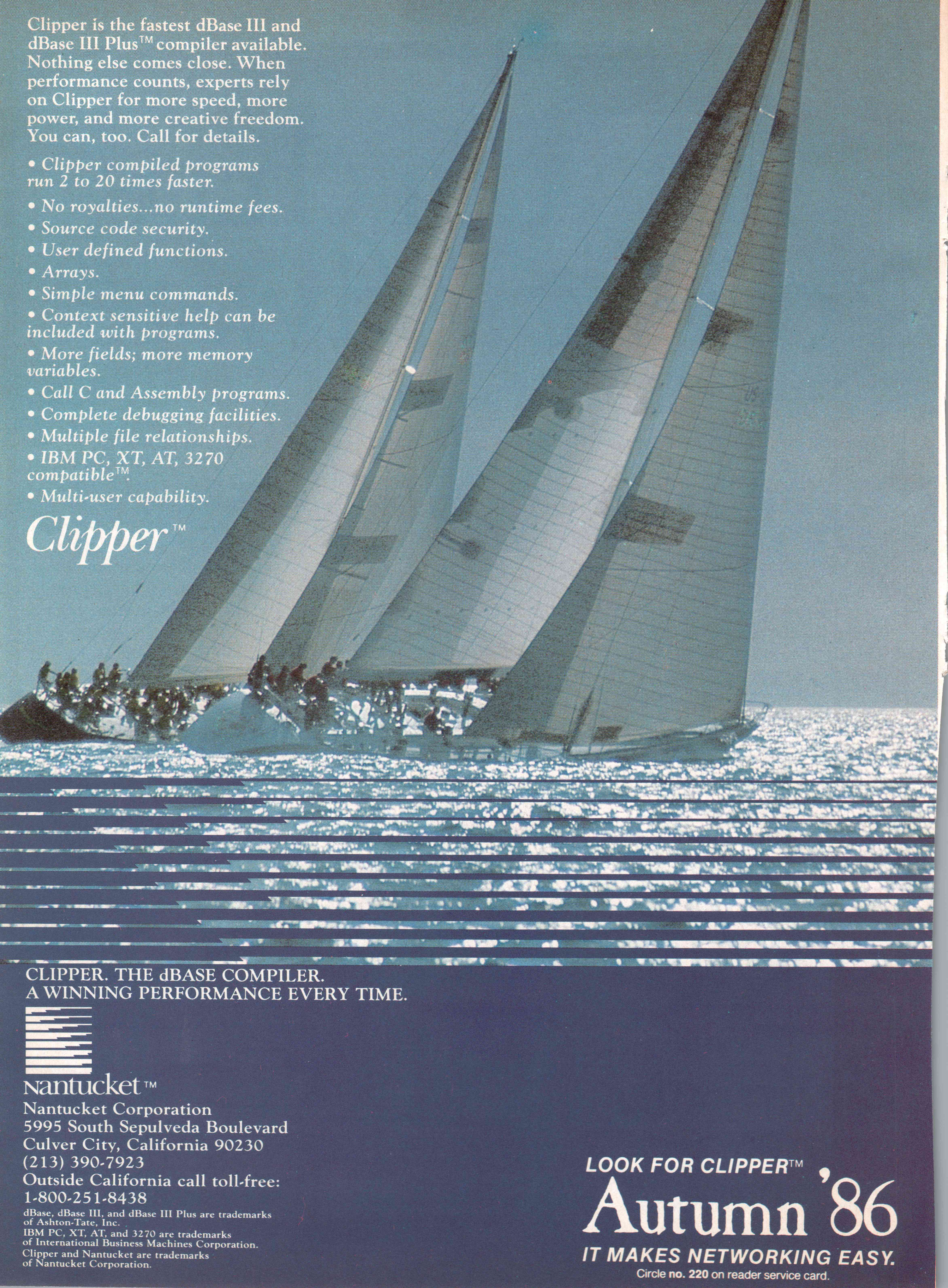
There's a lot of 32-bit talk in the press. Sampling quantities, blue sky and vaporware are fine, but maybe you'd rather not bet your company on them.

When you're planning to build the kind of leading-edge systems that win in the marketplace, Motorola offers you speed to market, aggressive pricing and the strength of a fully-realized 32-bit environment.

If you're an executive evaluating microprocessors, software and support, call us at 1-800-521-6274. We'll arrange a one-on-one management-level meeting. Fast. And we'll show you why a better microprocessor, coupled with here-and-now volume production, software and support, adds up to a winning combination.

UNIX is a registered trademark of AT&T
Concurrent DOS and GEMDOS is a trademark of Digital Research Inc.
VERSAdos is a registered trademark of Motorola Inc.
VRTX is a registered trademark of Hunter & Ready, Inc.
RTUX is a trademark of Emerge Systems
ADA is a trademark of the Department of Defense
VMEmodule and VMEsystem are trademarks of Motorola Inc.

MOTOROLA


Clipper is the fastest dBase III and dBase III Plus™ compiler available. Nothing else comes close. When performance counts, experts rely on Clipper for more speed, more power, and more creative freedom. You can, too. Call for details.
• Clipper compiled programs run 2 to 20 times faster.
• No royalties...no runtime fees.
• Source code security.
• User defined functions.
• Arrays.
• Simple menu commands.
• Context sensitive help can be included with programs.
• More fields; more memory variables.
• Call C and Assembly programs.
• Complete debugging facilities.
• Multiple file relationships.
• IBM PC, XT, AT, 3270 compatible™.
• Multi-user capability.
Clipper™
CLIPPER. THE dBASE COMPILER.
A WINNING PERFORMANCE EVERY TIME.
Nantucket™
Nantucket Corporation
5995 South Sepulveda Boulevard
Culver City, California 90230
(213) 390-7923
Outside California call toll-free:
1-800-251-8438
dBase, dBase III, and dBase III Plus are trademarks of Ashton-Tate, Inc.
IBM PC, XT, AT, and 3270 are trademarks of International Business Machines Corporation.
Clipper and Nantucket are trademarks of Nantucket Corporation.
LOOK FOR CLIPPER™
Autumn '86
IT MAKES NETWORKING EASY.
Circle no. 220 on reader service card.

apl language

APL*PLUS/PC *by STSC* *New Version*	595	449
APL*PLUS/PC Spreadsheet Mgr *by STSC*	195	139
APL*PLUS/PC Tools Vol 1 *by STSC*	295	199
APL*PLUS/PC Tools Vol 2 *by STSC*	85	59
APL*PLUS/UNX *For AT XENIX by STSC*	995	695
Btrieve *ISAM File Mgr by SoftCraft*	250	194
Financial/Statistical Library *by STSC*	275	195
Pocket APL *by STSC*	95	69
STATGRAPHICS *by STSC*	795	599

arity products

Arity Combination Package *New*	1225	1139
Expert System Development Pkg	295	259
File Interchange Toolkit	50	45
PROLOG Compiler & Interpreter	795	699
Screen Design Toolkit	50	45
SQL Development Package	295	259
Arity PROLOG Interpreter	350	309
Arity Standard Prolog	95	79

artificial intelligence

APES *by Prog Logic Sys*	395	339
APT *from Solution Systems* *New*	65	CALL
AutoIntelligence *by IntelligenceWare* *New*	990	CALL
ESP ADVISOR *by Expert Systems Intl*	895	839
PROLOG-2 Interface	395	369
ExpertEDGE Advanced *by Human Edge* *New*	2500	CALL
ExpertEDGE Professional *by Human Edge* *New*	5000	CALL
Experteach II *by IntelligenceWare*	475	379
EXSYS *Development Software by EXSYS*	395	319
First Class *by Human Edge*	495	399
GCLISP *Golden Common LISP by Gold Hill*	495	CALL
GCLISP 286 Developer *by Gold Hill*	1190	CALL
Insight 1 *by Level Five Research*	95	75
Insight 2+ *by Level Five Research*	485	379
Intelligence/Compiler *IntelligenceWare*	990	749
Logic-Line Series 1 *by Thunderstone*	90	85
Logic-Line Series 2 *by Thunderstone*	125	115
Logic-Line Series 3 *by Thunderstone*	150	139
LPA microPROLOG *by Prog Logic Sys*	99	89
with APES	149	129
LPA Professional microPROLOG	395	339
with APES	650	569
Compiler/Interpreter *New*	495	CALL
◆Microsoft LISP *Common LISP* *Sale*	250	159
PC Scheme *by Texas Instruments* *New*	95	85
Personal Consultant Easy *by TI* *New*	495	439
Personal Consultant Plus *by TI* *New*	2950	2599
Personal Consultant Runtime *New*	CALL	CALL
PROLOG-2 Interpreter *by ESI*	450	419
PROLOG-2 Interpreter and Compiler	895	839
QNIAL *by NIAL Systems*	375	349
TransLISP *from Solution Systems* *New*	95	CALL
Turbo PROLOG Compiler *by Borland Intl*	100	75

assembly language

386 ASM/LINK *Cross Asm by Phar Lap*	495	CALL
8088 Assembler *by 2500 AD*	100	89
ASMLIB *Function Library by BC Assoc*	149	129
asmTREE *B-Tree Dev System by BC Assoc* *New*	395	339
Cross Assemblers *Various 2500 AD*	CALL	CALL
◆Microsoft Macro Assembler *Sale*	150	89
Norton Utilities *by Peter Norton*	100	69
Turbo EDITASM *by Speedware*	99	84
UniWare Cross Assemblers *Various* *New*	295	CALL
Visible Computer: 8088 *Software Masters*	80	65

basic language

BetterBASIC *by Summit Software*	200	129
8087 Math Support	99	75
Btrieve Interface	99	75
C Interface	99	75
Run-time Module	250	169
EXIM Services Toolkit *by EXIM* *New*	CALL	CALL
Finally *by Komputerwerks* *New*	99	85
Inside Track *from Micro Help*	65	55
MACH 2 *by Micro Help*	75	65
◆Microsoft QuickBASIC *Sale*	99	59
Peeks 'n Pokes *from MicroHelp*	45	39
Professional BASIC *by Morgan*	99	75
8087 Math Support	50	42
Stay-Res *by MicroHelp* *New*	95	85
True Basic *w/BASICA Converter* *New Version*	200	105
True Basic *w/Converter & Run-time*	295	199
Advanced String Library	50	45
Asynch Communication Support	50	45
BASICA Converter	50	45
Btrieve Interface	50	45
Developer's Toolkit	50	45
Formlib	50	45
Hercules Graphic Support	50	45
Run-time Module	150	109
Sorting & Searching	50	45

blaise products

ASYNCH MANAGER *Specify C or Pascal*	175	135
C TOOLS	125	99
C TOOLS 2	100	79
C TOOLS PLUS	175	135
EXEC *Program Chainer*	95	75
PASCAL TOOLS	125	99
PASCAL TOOLS 2	100	79
PASCAL TOOLS & PASCAL TOOLS 2	175	135
RUNOFF *Text Formatter*	50	45
TURBO ASYNCH PLUS	100	83
TURBO POWER TOOLS PLUS	100	83
VIEW MANAGER *Specify C or Pascal*	275	199

borland products

REFLEX *Data Base System*	150	99
REFLEX Workshop	70	48
REFLEX & REFLEX Workshop	200	129
Turbo DATABASE TOOLBOX	70	48
Turbo EDITOR TOOLBOX	70	48
Turbo GAMEWORKS TOOLBOX	70	48
Turbo GRAPHIX TOOLBOX	70	48
Turbo LIGHTNING	100	65
◆Turbo PASCAL *with 8087 and BCD*	100	65
Turbo Prolog Compiler	100	65
Turbo TUTOR *for Turbo PASCAL*	40	28
Word Wizard	70	48
Word Wizard and Turbo Lightning	150	99

c++

C++ *from Guidelines* *New*	195	179

c compilers

C-86 *by Computer Innovations*	395	279
Datalight C Compiler *Small Model*	60	49
Datalight Developer Kit *w/Large Model*	99	79
DeSmet C *w/Debugger*	159	145
DeSmet C *w/Debugger & Large Case*	209	193
Eco-C *Development System by Ecosoft*	125	89
Lattice C Compiler *from Lattice*	500	294
Mark Williams Let's C	75	58
with csd *Source Debugger*	150	109
Mark Williams MWC-86	495	289
◆Microsoft C *with CodeView* *Sale*	450	268
UniWare 68000/10/20 Cross Compiler *New*	595	CALL
Wizard C Combo *by Wizard Systems* *New*	750	599
Wizard C Compiler	450	359
ROM Development Pkg *New*	350	299

c interpreters

C-terp *by Gimpel, Specify compiler*	300	235
C Trainer *with Software by Catalytix*	122	CALL
Instant C *by Rational Systems*	500	CALL
Introducing C *by Computer Innovations*	125	104
Run/C *from Lifeboat*	150	89
Run/C Professional *from Lifeboat*	250	169

c utilities

APT *by Shaw American Technology*	395	299
Basic C Library *by C Source*	175	129
C Essentials *by Essential Software*	100	CALL
C-ISAM *by Informix*	225	195
C to dBase *by Computer Innovations*	150	135
c-tree *ISAM File Manager by FairCom*	395	329
r-tree *Report Generator* *New*	295	249
c-tree & r-tree Combo Package *New*	650	529
C Utility Library *Essential* *New Version*	185	135
C Windows *by Syscom*	100	89
C Wings *by Syscom*	50	45
CI ROMPac *by Computer Innovations*	195	139
dbQUERY *by Raima*	195	155
dbVISTA *Single-User DBMS by Raima*	195	155
with Source Code	495	425
dbVISTA *Multi-User DBMS by Raima*	495	425
with Source Code	990	845
dBx *dBase/C Translator by Desktop AI*	350	325
Entelekon Combo Package	200	169
C Function Library	130	109
C Windows	130	109
Superfonts for C	50	43
Essential Comm Library *with Debugger* *New*	250	195
Breakout Debugger *Any language* *New*	125	99
Essential Comm Library *New*	185	135
Essential Graphics *by Essential Software*	250	195
Flash-up Windows *by Software Bottling*	90	79
GraphiC *Mono v2.2 by Sci Endeavors*	280	209
GraphiC *Color v3.0 by Sci Endeavors*	350	289
GRAFLIB *by The Librarian*	175	CALL
Greenleaf Comm Library *by Greenleaf*	185	134
Greenleaf Data Windows *by Greenleaf* *New*	225	189
with Source Code *New*	395	329
Greenleaf Functions *by Greenleaf*	185	134
The HAMMER *by OES Systems*	195	149
HALO *by Media Cybernetics*	300	209
HELP/Control *by MDS*	125	109
◆MetaWINDOWS *No Royalties*	185	115
MetaFONTS	80	58
◆MetaWINDOWS/Plus *by Metagraphics*	235	189
MetaFONTS/Plus	235	189
On-line Help *from Opt-Tech Data Proc*	149	109
PANEL *by Roundhill Computer Systems*	295	224
PC Lint *by Gimpel Software*	139	105
PLOTHI *by The Librarian*	175	CALL
PLOTHP *by The Librarian*	175	CALL
Sci Subroutine Library *by Peerless*	175	139
Vector87 *by Vectorplex Data Systems* *New*	120	109
Vitamin C *by Creative Programming*	150	135
VC Screen *Forms Designer*	100	84
Zview *by Data Management Consultants*	245	189

cobol language

Micro Focus COBOL Workbench	4000	3379
Micro Focus Level II COBOL	1500	549
COGRAPHICS	250	199
COMATH	200	159
FORMS-2	300	259
Level II Animator	900	349
Level II SOURCEWRITER	2000	CALL
Micro Focus Level II COBOL *for Novell*	2000	1699
Micro Focus Micro/SPF	175	149
Micro Focus Professional COBOL	3000	2295
Multi-user Runtime *for PC Network*	500	429
◆Microsoft COBOL *Sale*	700	419
◆Microsoft COBOL Tools *w/Debugger* *Sale*	350	199
Realia COBOL *New Version*	995	785
RM/COBOL *by Ryan-McFarland*	950	639
RM/COBOL 8X *ANSI 85 by Ryan-McFarland*	1250	895

debuggers & profilers

386 DEBUG *Cross Debugger by Phar Lap*	175	139
Advanced Trace-86 *by Morgan Computing*	175	125
CI Probe *by Computer Innovations*	225	189
Codesifter *Profiler by David Smith*	119	98
Codesmith-86 *by Visual Age*	145	108
DSD86 *by Soft Advances*	70	65
DSD87 *by Soft Advances*	100	89
Periscope I *by Data Base Decisions*	295	245
Periscope II *w/NMI Breakout Switch*	145	109
Periscope II-X *Software only*	115	84
The PROFILER *with Source Code by DWB*	125	94
The WATCHER *Profiler by Stony Brook*	60	55

forth language

CFORTH *Native Code Compiler by LMI*	300	239
Forth/83 Metacompiler *Specify Target*	750	599
PC/Forth *by Laboratory Microsystems*	150	119
PC/Forth+ *by Laboratory Microsystems*	250	209
Advanced Color Graphics Support	100	79
Enhanced Graphics Support	200	159
Intel 8087 Support	100	79
Interactive Symbolic Debugger	100	79
Native Code Optimizer	200	159
PCTERM *Modem Pgm for Smartmodem*	100	79
Software Floating Point	100	79
UR/Forth *by Laboratory Microsystems* *New*	350	279
Object Module Libraries *New*	500	395
Source Code License *New*	1500	995

fortran language

50 MORE: FORTRAN *by Peerless Engr*	125	99
◆ACS Time Series *Alpha Computer Service*	495	419
Btrieve *ISAM File Mgr by SoftCraft*	250	194
Essential Graphics *by Essential Software*	250	195
◆For-Winds *Alpha Computer Service*	90	78
◆Forlib-Plus *Alpha Computer Service*	70	54
FORTLIB *by The Librarian*	95	CALL
FORTRAN Addenda *by Impulse Engr*	95	89
FORTRAN Addendum *by Impulse Engr*	165	149
GRAFLIB *by The Librarian*	175	CALL
HALO *by Media Cybernetics*	300	209
I/O PRO *w/No Limit Library by MEF*	390	349
Microcompatibles Combo Package	240	219
Grafmatic	135	119
Plotmatic	135	119
◆Microsoft FORTRAN *Sale*	350	199
No Limit *by MEF Environmental*	129	115
PANEL *Screen Designer by Roundhill*	295	224
PLOTHI *by The Librarian*	175	CALL
PLOTHP *by The Librarian*	175	CALL
◆RM/FORTRAN *Ryan-McFarland*	595	389
Sci Subroutine Library *by Peerless*	175	138
◆Statistician *Alpha Computer Service*	295	249
◆Strings & Things *Alpha Computer Service*	70	54
Vector87 *by Vectorplex Data Systems* *New*	120	109

gss products

GSS Graphics Development Toolkit	495	375
GSS Kernel System *for DOS*	495	375
GSS Kernel System *for IBM RT*	795	649
GSS Metafile Interpreter	295	239
GSS Plotting System	495	375
GSS SOLUTIONS Chart	295	239
GSS SOLUTIONS Plottalk	295	239
GSS SOLUTIONS Terminal	295	239

lattice products

Lattice C Compiler *from Lattice*	500	294
with Library Source Code	900	545
C Cross Reference Generator	50	39
with Source Code	200	149
C-Food Smorgasbord *Function Library*	150	98
with Source Code	300	188

◆◆◆◆◆◆◆◆◆ These products have sale prices or special offers that are available until 12/31/86. Turn to the first page of this ad for more information.

Circle no. 129 on reader service card.

C-Sprite *Source Level Debugger*	175	138
Curses *Screen Manager*	125	94
with Source Code	250	184
dBC *dBase File Manager for C*	250	189
with Source Code	500	378
LMK *Make Facility*	195	145
RPG II Compiler *No Royalties*	750	635
RPG II Combo *with SEU & Sort/Merge* New	1100	939
SecretDisk *File Encryption Utility*	120	94
SideTalk *Resident Communications*	120	94
Text Management Utilities	120	94
TopView Toolbasket *Function Library*	250	189
with Source Code	500	378
Z-80 C Cross Compiler	500	378
with Library Source Code	1000	749

logitech products

LOGIMOUSE C7 *Mouse Hardware*	99	85
with PLUS Pkg	119	99
with PLUS Pkg & PC Paintbrush	169	145
with PLUS Pkg & CAD Software	189	159
with PLUS Pkg & CAD & Paint New	219	189
MODULA-2/86 Holiday Package New	199	159
MODULA-2/86 Compiler	89	63
MODULA-2/86 with 8087 Support	129	103
MODULA-2/86 with PLUS Pkg	189	147
Library Sources	99	89
Make Utility	29	27
ROM Package	199	179
Run Time Debugger	69	59
Turbo to Modula Translator	49	45
Utilities Package	49	45
Window Package	49	45
REPERTOIRE *for MODULA-2/86 by PMI* New	89	79

microport products

System V/AT *by Microport Systems* New	440	395
RUNTIME SYSTEM *(Operating Sys)* New	160	145
SOFTWARE DEVELOPMENT SYSTEM New	170	155
TEXT PREPARATION SYSTEM New	170	155
USER UPGRADE *3 to 8 Users* New	100	85

microsoft products

◆ Microsoft BASIC *for XENIX* Sale	350	219
◆ Microsoft C *with CodeView* Sale	450	268
◆ Microsoft COBOL Compiler Sale	700	419
◆ *for XENIX* Sale	995	599
◆ Microsoft COBOL Tools *with Debugger* Sale	350	199
◆ *for XENIX* Sale	450	289
◆ Microsoft FORTRAN Compiler Sale	350	199
◆ *for XENIX* Sale	695	429
◆ Microsoft Learning DOS Sale	50	35
◆ Microsoft LISP *Common LISP* Sale	250	159
◆ Microsoft MACH 10 *Bundled Pkg* Sale	549	379
◆ Microsoft MACH 10 *Board* Sale	399	279
◆ Microsoft Macro Assembler Sale	150	89
◆ Microsoft Mouse *Bus Version* Sale	175	119
◆ Microsoft Mouse *Serial Version* Sale	195	129
◆ Microsoft muMath *Includes muSIMP* Sale	300	179
◆ Microsoft Pascal Compiler Sale	300	179
◆ *for XENIX* Sale	695	429
◆ Microsoft QuickBASIC Sale	99	59
◆ Microsoft Sort Sale	195	129
◆ Microsoft Windows Sale	99	59
◆ Microsoft Windows Development Kit Sale	500	299

other languages

CCS MUMPS *Single-User by MGlobal*	60	55
CCS MUMPS *Multi-User by MGlobal*	450	379
Janus/ADA C Pack *by R&R Software*	95	89
Janus/ADA D Pack *by R&R Software*	900	795
Methods *Smalltalk by Digitalk*	79	68
Personal REXX *by Mansfield Software*	125	109
Smalltalk/V *by Digitalk*	99	88
Smalltalk/Comm New	49	45
SNOBOL4+ *by Catspaw*	95	84

other products

Dan Bricklin's Demo Pgm *Software Garden*	75	59
FASTBACK *by 5th Generation Systems*	179	149
Informix *for DOS by Informix*	795	639
Informix4GL *for DOS by Informix*	995	799
InformixSQL *for DOS by Informix*	795	639
Instant Replay *by Nostradamus* New	90	79
Interactive EASYFLOW *by Haventree*	150	129
MKS Toolkit *with vi by MKS*	139	119
Norton Commander *by Peter Norton* New	75	55
OPT-Tech Sort *by Opt-Tech Data Proc*	149	115
PrintQ *by Software Directions*	89	84
Quilt Computing Combo Package	199	169
QMake *Program Rebuild Utility*	99	84
SRMS *Software Revision Mgmt Sys*	125	109
screenplay *all varieties by Flexus* New	CALL	CALL
SoftScreen/HELP *by Dialectic Systems* New	195	149
Source Print *by Aldebaran Labs*	139	115

LOWEST PRICES

Since this ad is prepared in advance of publication, some of our current prices may be lower than what's advertised here. Call for latest pricing.

FREE SHIPPING

Orders within the USA (including Alaska Hawaii) are shipped FREE via UPS. Express shipping is available at the shipping carrier's standard rate with no rush fees or handling charges. To avoid delays when ordering by mail, please call first to determine the exact cost of express shipping.

CREDIT CARDS

VISA and MasterCard are accepted at no extra cost. Your card is charged when your order is shipped. Mail orders please include credit card expiration date and telephone number.

CODs AND POs

CODs and Purchase Orders are accepted at no extra cost. POs with net 30-day terms are available to qualified US accounts only.

FOREIGN ORDERS

Shipping charges for foreign and Canadian orders are based on the shipping carrier's standard rate. Since rates vary between carriers, please call or write for the exact cost. Foreign orders (except Canada), please include an additional $10 for customs form preparation. All payments must be made with US funds drawn on a US bank. Please include your telephone number when ordering by mail. Due to government regulations, we cannot ship to all countries.

VOLUME ORDERS

Call for special pricing.

SOUND ADVICE

Our knowledgeable technical staff can compare products, answer technical questions and send you detailed product information tailored to your needs.

30-DAY GUARANTEE

Most of our products come with a 30-day return guarantee or a 30-day evaluation period. Please note that some products are restricted by their manufacturers from this guarantee. Call for more information.

CALL TOLL FREE

US	800-336-1166
CANADA	800-225-1166
OHIO	216-877-3781
CUSTOMER SERVICE	216-877-1110

Hours: Weekdays 8:30 AM to 8:00 PM EST.
Ohio customers please add 5% state sales tax.
Call or write for our FREE comprehensive price guide.
Prices are subject to change without notice.

136 SUNNYSIDE ST. HARTVILLE, OHIO 44632

Taskview *by Sunny Hill Software*	80	65
TLIB *by Burton Systems Software* New	100	89
VTEK *Term Emulator by Sci Endeavors* New	150	129

phoenix products

◆ Pasm86 *Macro Assembler* Sale	195	109
◆ Pdisk *Hard Disk & Backup Utility* Sale	195	109
◆ Pfantasy Pac *Phoenix Combo* Sale	1295	829
◆ Pfinish *Performance Analyzer* Sale	395	219
◆ Pfix-86 Plus *Symbolic Debugger* Sale	395	219
◆ PforCe *Comprehensive C Library* Sale	395	219
◆ Plink-86 Plus *Overlay Linker* Sale	495	299
◆ Pmaker *Make Utility* Sale	125	79
◆ Pmate *Macro Text Editor* Sale	195	109
◆ Pre-C *Lint Utility* Sale	295	149
◆ Ptel *Binary File Transfer Program* Sale	195	109

polytron products

PolyBoost *The Software Accelerator* New	80	69
Polytron C Beautifier	49	45
Polytron C Library I	99	78
Polytron PowerCom *Communications*	179	139
PolyLibrarian *Library Manager*	99	78
PolyLibrarian II *Library Manager*	149	115
PolyMake *UNIX-like Make Facility*	99	78
PolyWindows Products *All Varieties*	CALL	CALL
PolyXREF *Complete Cross Ref Utility*	219	179
PolyXREF *One language only*	129	109
PVCS *Version Control System*	395	325

softcraft products

Btrieve *ISAM Mgr with No Royalties*	250	194
Xtrieve *Query Utility* New Version	245	CALL
Rtrieve Report Option New Version	145	CALL
Btrieve/N *for Networks*	595	464
Xtrieve/N New Version	595	CALL
Rtrieve/N Report Option New Version	345	CALL

text editors

Brief *from Solution Systems*	195	CALL
Epsilon *Emacs-like editor by Lugaru*	195	159
KEDIT *by Mansfield Software*	125	105
PC/VI *by Custom Software Systems*	149	129
SPF/PC *by Command Technology Corp*	195	149
Vedit *by CompuView*	150	109
Vedit Plus *by CompuView*	185	139

turbo pascal utilities

ALICE *Interpreter by Software Channels*	95	68
Btrieve *ISAM File Mgr* See SoftCraft	250	194
FirsTime for Turbo *by Spruce Tech*	75	59
Flash-up Windows *by Software Bottling*	90	79
HELP/Control *by MDS*	125	109
On-line Help *from Opt-Tech Data Proc*	149	109
Report Builder *by Royal American* New	75	CALL
Screen Sculptor *by Software Bottling*	125	94
System Builder *by Royal American* New	100	CALL
TDebugPLUS *by TurboPower Software*	60	53
Turbo EXTENDER *by TurboPower Software*	85	68
Turbo Professional *by Sunny Hill*	70	49
TurboHALO *from IMSI*	129	99
TurboPower Utilities *by TurboPower*	95	84
TurboRef *by Gracon Services*	50	45
◆ TurboSmith *Visual Age Debugger*	58	45
◆ TurboWINDOW *by MetaGraphics*	80	65

wendin products

Operating System Toolbox	99	84
PCUNIX *Operating system*	99	84
PCVMS *Similar to VAX/VMS*	99	79
XTC *Text editor with Pascal source*	99	79

xenix system v

See also Microport System V/AT section.

XENIX System V Complete System *by SCO*	1295	999
XENIX Development System	595	499
XENIX Operating Sys *Specify XT/AT*	595	499
XENIX Text Processing Package	195	149

xenix products

APL*PLUS/UNX *For AT XENIX by STSC*	995	695
Btrieve *ISAM File Mgr by SoftCraft*	595	464
C-ISAM *by Informix*	319	285
c-tree *ISAM Mgr w/Source by FairCom*	395	329
dBx *dBase/C Translator by Desktop AI*	550	499
dbVISTA *Single or Multi User by Raima*	CALL	CALL
Informix *by Informix*	995	795
Informix4GL *by Informix*	1500	1199
InformixSQL *by Informix*	995	795
Lyrix *by Informix*	595	449
Micro Focus Level II Compact COBOL	1000	795
Forms-2	400	319
Level II ANIMATOR	600	479
◆ Microsoft Languages	CALL	CALL
Networks for XENIX *by SCO*	595	495
PANEL *Screen Designer by Roundhill*	750	549
RM/COBOL *by Ryan-McFarland*	1250	949
◆ RM/FORTRAN *by Ryan-McFarland*	750	549
SCO Professional *Lotus clone by SCO*	795	595

programmer's connection

Circle no. 129 on reader service card.

Listing Four (Listing continued, text begins on page 16.)

```
                pop ds          ; restore ds
                iret            ; return from interrupt

; system interlock

tick:           mov t_astrm,0FFh ; terminate application task
                sys_entr tmrflg ; enter system state

; set machine environment

                sys_sctx        ; save processor context
                push bp         ; protect bp
                mov bp,sp       ; mark stack location
                lea ax,tmr_xtm  ; accumulate psuedo time
                push ax
                call t_rtmark
                mov sp,bp
                mov tmrpxtm,ax  ; store prior pointer

; real time system processing

                inc tmr_dct     ; remove display harmonics
                mov al,DSPCT
                xor al,tmr_dct
                jnz tick4
                mov tmr_dct,al
                push tmr_dspl   ; display physical window
                call w__cdspl
                mov sp,bp       ; restore stack pointer
                inc tmr_ict     ; increment interrupt counter
                inc tmr_tkct    ; increment tick clock

; time base generation

                mov ax,ISKP0    ; long term time base correction
                xor ax,tmr_ict
                jnz tick1
                mov tmr_ict,ax
                call tick5      ; update system tick clock
tick1:          inc tmr_idv0    ; generate clock time base
                mov al,IDV0
                xor al,tmr_idv0
                jnz tick3
                mov tmr_idv0,al
                call tick5      ; update system tick clock
                inc tmr_idv1    ; primary alternate time base correction
                mov al,ISKP1
                xor al,tmr_idv1
                jnz tick2
                mov tmr_idv1,al
                int RLCINT      ; update alternate time base
                inc tmr_idv2    ; secondary alternate time base correction
                mov al,ISKP2
                xor al,tmr_idv2
                jnz tick2
                mov tmr_idv2,al
                int RLCINT      ; update alternate time base
tick2:          int RLCINT      ; update alternate time base

; terminate interrupt service

tick3:          push tmrpxtm    ; restore original psuedo time accumulator
                call t_rtmark
                mov sp,bp
                pop bp          ; restore bp
                test tmrflg,0FFh ; test for interrupted system task
                jnz tick4
                xor ax,ax       ; terminate task
                mov tmr_ilck,al ; enable reentrance
                retn            ; near return to system task management

tick4:          sys__rctx       ; restore processor context
                cli             ; interrupts off
                mov tmr_ilck,0  ; enable reentrance
                pop ds          ; restore ds
                iret            ; return to interrupted task

; update system tick counter

tick5:          mov ax,td_tct   ; test for no overflow
                inc ax
                cmp ax,td_ref
                jne tick6
                call td__upd    ; update clock
                xor ax,ax       ; reset tick counter
                mov td_ref,ax
                mov td_tct,ax
tick6:          inc td_tct      ; increment tick counter
                retn            ; return

t__tick         endp

comment ~
tmr__int                        initialize timer

tmr__int()

All data areas necessary for clock maintenance are initialized. The hardware
timer is programmed for the appropriate rate and its interrupt vector is made
to point to sys_tmr. The original vector is relocated and will be used by
sys_tmr as the alternate time base.

~

tmr__int  proc near
          call tmr__dsi  ; diable interrupts
```

(continued on page 76)

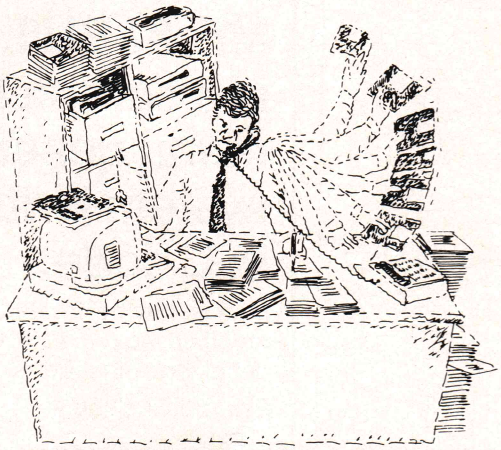
Listing Four (Listing continued, text begins on page 16.)

```
          mov ax,dgroup  ; set data segment
          mov sys_dgrp,ax
          mov ax,cs      ; set code segment
          lea si,sys_stat
          mov [si].rcs,ax
          cli            ; interrupts off
          mov tmr_ilck,0FFh ; lockout t__tick
          mov bx,tmr_sync ; test for no synchronization function
          test bx,bx
          jz int0
          lea bx,tmr_sync ; synchronize timer interrupt
          call [bx]
          jmp short int1 ; continue
int0:     call tmr__tmr  ; start timer
int1:     call t_rdclk   ; set real time clock phase
          mov t_rttick,ax
          mov t_rtrfct,ax ; set reference count
          mov t_rtactg,offset dgroup:t_syxtm ; initialize time accumulator
          call td__set   ; set current time
          sti            ; interrupts on
          xor ax,ax      ; form 0
          push ds        ; protect ds
          mov ds,ax      ; relocate original interrupt vector
          mov di,ax
          cli
          mov ax,[di+4*TMRINT]
          mov [di+4*RLCINT],ax
          mov ax,[di+4*TMRINT+2]
          mov [di+4*RLCINT+2],ax
          mov ax,offset pgroup:t__tick ; set interrupt service
          mov [di+4*TMRINT],ax
          mov ax,cs
          mov [di+4*TMRINT+2],ax
          sti            ; interrupts on
          pop ds         ; restore ds
          call tmr__eni  ; enable interrupts
          ret            ; return
tmr__int  endp


comment ~
tmr__clr                 reset time base generation

tmr__clr()

The time base adjustment variables are reset. This function is to be called by
td__set when the time of day is initially set from a continuous clock. Nothing
is returned.
~

tmr__clr  proc near
          xor ax,ax      ; zero time base generation variables
          mov tmr_idv0,al
          mov tmr_idv1,al
          mov tmr_idv2,al
          ret            ; return
tmr__clr  endp


comment ~
tmr__rst                 reset timer

tmr__rst()

The original interrupt service routine is restored and the hardware clock is
reprogrammed. However, the original hardware values are not available in this
edition. Therefore the original system state cannot always be restored.


tmr__rst  proc near
          mov tmr_ilck,0FFh ; lock out interrupt service
          push ds        ; protect ds
          xor ax,ax      ; restore original interrupt vector
          mov ds,ax
          mov di,ax
          call tmr__dsi  ; disable timer interrupt
          cli            ; interrupts off
          mov ax,[di+4*RLCINT]
          mov [di+4*TMRINT],ax
          mov ax,[di+4*RLCINT+2]
          mov [di+4*TMRINT+2],ax
          pop ds         ; restore ds
          xor bx,bx      ; restart hardware timer
          call tmr__str
          sti            ; interrupts on
          call tmr__eni  ; enable timer interrupt
          ret            ; return
tmr__rst  endp

comment ~
tmr__tmr                 restart hardware timer

tmr__tmr\\

Channel 0 of an 8253 timer is initialized to mode 3. The count in bx is then
programmed.


tmr__tmr  proc near      ; restart timer
          mov bx,tmr_dvsr ; set tele system tick period
tmr__str  proc near      ; set timer period
          mov al,20      ; reset 8253 (mode 0, count >= 8,192)
          out TMRPRT+3,al ; [> 6.8 msec]
          iowait
          out TMRPRT,al
          mov al,36h     ; initialize 8253 (mode 3, both bytes)
          iowait
```

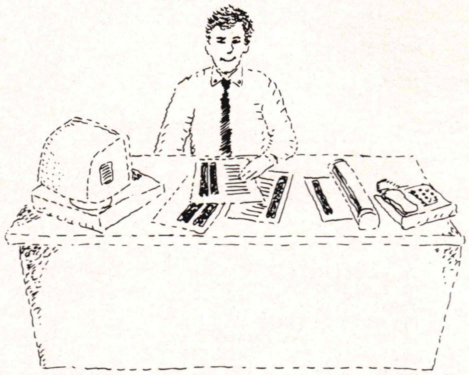
```
        out TMRPRT+3,al
        mov al,bl
        iowait
        out TMRPRT,al
        mov al,bh
        iowait
        out TMRPRT,al
        ret             ; return
tmr__str endp
tmr__tmr endp

comment ~
tmr__sts                read timer status

tmr__sts()

The returned value is the current count in the timer.
~

tmr__sts proc near      ; read timer status
        mov al,00h      ; set read mode
        out TMRPRT+3,al
        nop             ; allow timer chip to recover
        in al,TMRPRT    ; read count
        mov ah,al
        in al,TMRPRT
        xchg ah,al
        ret             ; return
tmr__sts endp

comment ~
tmr__dsi                disable interrupt

tmr__dsi()

The timer interrupt is disabled at the 8259 interrupt controller.
~

tmr__dsi proc near
        cli             ; interrupts off
        in al,INTPRT+1  ; disable timer interrupt
        or al,TMRMSK
        iowait
        out INTPRT+1,al
        sti             ; interrupts on
        ret             ; return
tmr__dsi endp

comment ~
tmr__eni                enable interrupt

tmr__eni()

The timer interrupt is enabled at the 8259 interrupt controller.
~

tmr__eni proc near
        cli             ; interrupts off
        in al,INTPRT+1; ; enable timer interrupt
        and al,not TMRMSK
        iowait
        out INTPRT+1,al
        sti             ; interrupts on
        ret             ; return
tmr__eni endp

comment ~
t_rdclk                 read real time clock

t_rdclk()

The current value of the real time clock is read and returned.
~

DMAREG  equ 0           ; refresh address DMA register

t_rdclk proc near
rdclk0: mov dx,DMAREG   ; set DMA register address
        call t__rdclk   ; read time
        mov bx,ax       ; store time
        call t__rdclk   ; read time again
        cmp ah,bh       ; test for interruption
        jne rdclk0
        ret             ; return
t_rdclk endp

t__rdclk proc near
        cli             ; interrupts off
        in al,dx        ; read time
        mov ah,al
        iowait
        in al,dx
        xchg al,ah
        sti             ; interrupts on
        ret             ; return
t__rdclk endp

comment ~
t_rtmark                mark execution interval

t_rtmark(np)
pointer *np;

The number of refreshes since the last call to t_rtmark is accumulated. Then
the reference count is reset. np points to the area that will accumulate the
number of refreshes to the next call. The returned value is the original
accumulator pointer.

t_rtmark proc near
        push bp         ; protect bp
        mov bp,sp       ; establish parameter addressability
```

(continued on page 80)

SOFTSTRIP®
NOW OFFERS YOU SOMETHING NEVER BEFORE AVAILABLE...



CONVENTIONAL DATA HANDLING



THE SOFTSTRIP SYSTEM

A CHOICE.

Until now you were stuck with disks.

No more. Install our unique STRIPPER™ software on your personal computer today and discover the many benefits of the fastest, easiest, least expensive way to handle information.

STRIPPER lets you print – ON PAPER – your own machine readable Softstrip data strips using your dot matrix printer. The Softstrip System Reader reads that information into a computer rapidly. With STRIPPER and the reader, your PC and printer become part of the most versatile information handling system available.

With this system you can do anything you wish with any data you have in your PC – ON PAPER.

DATA ENTRY: Why use keystrokes when you can eliminate them with data strips? Whatever the document - invoices, packing slips, memos, letters, sales reports, the list is endless — simply print a data strip right on the same printed page. Now you have a document that is both human readable and machine readable. A typical document can be entered in only 15 seconds using data strips. And, it ends keystroke errors forever.

DATA DISTRIBUTION: Why copy disks? It's time consuming and expensive. Softstrip data strips will end all that. Simply photocopy as many data strips as you like and send them by mail. Data strips ignore folding, coffee stains, ink marks and, by the way, magnetic fields. And if you're using telecommunications, you can stop making the phone company rich.

DATA STORAGE AND RETRIEVAL: Why have a file of disks and a file of paper? Eliminate one with Softstrip data strips. File the data strip with the document. Better still, print the strip right on the document. Then put it in a file or binder.

Retrieval is simple. To find existing data, pull the document and its related data strip from the file. They've been stored together. Then use the reader to enter the data. No more hassle trying to match documents with the right disk — if you can find it.

DATA TRANSFER: Why bother with cables, modems and phone lines to move files between computers? A Softstrip data strip generated by an IBM PC can be read into another PC, or compatible, an Apple or even a Macintosh. If you work at home on a Macintosh, make a data strip on your printer, take it into the office and read it into your IBM PC. Simple. And we've created the utilities to let you do that easily. (See Application Notes on opposite page.)

Fascinating, isn't it? Anything you can do with disks can be done with the Softstrip data strip system — faster, easier and at lower cost — ON PAPER.

All you need is STRIPPER software at **$19.95** and the Softstrip System Reader at **$199.95.**

Listing Four (Listing continued, text begins on page 16.)

```
            call tmr__dsi  ; disable timer interrupt
            call t_rdclk   ; read real time clock
            mov bx,ax      ; protect current count
            xchg bx,t_rtrfct ; update reference count
            sub ax,bx      ; compute execution interval
            jnc mark1      ; test for no overflow
            neg ax         ; adjust count
mark1:      mov bx,t_rtactg ; accumulate execution time
            add ax,[bx]
            mov [bx],ax
            jnc markxit
            add word ptr [bx+2],1
            jnc markxit
            inc word ptr [bx+4]
markxit:    mov ax,bx      ; return orginal pointer
            mov bx,[bp].t_np0 ; set new accumulator pointer
            mov t_rtactg,bx
            call tmr__eni  ; enable timer interrupt
            pop bp         ; restore bp
            ret            ; return
t_rtmark    endp

comment ~
w__cdspl                  display physical buffer

w__cdspl(pw)
struct w_phys *pw;

Physical window pw is displayed.  This function is called by the system tick
clock interrupt service function.  Nothing is returned.
~

w__cdspl    proc near
            ret            ; return
w__cdspl    endp

comment ~
w__sync                   synchronize interrupt to display

w__sync()

The system tick clock timer is adjusted so that w__dsply executes just prior
to the vertical blanking interval.  Nothing is returned.
~

w__sync     proc near
            call tmr__tmr  ; start timer
w__sync     endp

comment ~
td__set                   set time of day clock

td__set()

The clock is set to the current time.  Nothing is returned.
~

td__set     proc near
            ret            ; return
td__set     endp

comment ~
td__upd                   update clock

td__upd()

The clock is updated based on the number of ticks since it was last updated.
The normal return (ax) is _F.  _E is returned if the call was locked out.
~

td__upd     proc near
            ret            ; return
td__upd     endp

            endps

            end
```

End Listings

SAS Institute Inc. Announces

Lattice C Compilers for Your IBM Mainframe

Two years ago...

SAS Institute launched an effort to develop a subset of the SAS® Software System for the IBM Personal Computer. After careful study, we agreed that C was the programming language of choice. And that the Lattice® C compiler offered the quality, speed, and efficiency we needed.

One year ago...

Development had progressed so well that we expanded our efforts to include the entire SAS System on a PC, written in C. And to insure that the language, syntax, and commands would be identical across all operating systems, we decided that all future versions of the SAS System—regardless of hardware—would be derived from the same source code written in C. That meant that we needed a C compiler for IBM 370 mainframes. And it had to be good, since all our software products would depend on it.

So we approached Lattice, Inc. and asked if we could implement a version of the Lattice C compiler for IBM mainframes. With Lattice, Inc.'s agreement, development began and progressed rapidly.

Today...

Our efforts are complete—we have a first-rate IBM 370 C compiler. And we are pleased to offer this development tool to you. Now you can write in a single language that is source code compatible with your IBM mainframe and your IBM PC. We have faithfully implemented not only the language, but also the supporting library and environment.

Features of the Lattice C compiler for the 370 include:

- **Generation of reentrant object code.** Reentrancy allows many users to share the same code. Reentrancy is not an easy feature to achieve on the 370, especially if you use non-constant external variables, but we did it.
- **Optimization of the generated code.** We know the 370 instruction set and the various 370 operating environments. We have over 100 staff years of assembler language systems experience on our development team.
- **Generated code executable in both 24-bit and 31-bit addressing modes.** You can run compiled programs above the 16 megabyte line in MVS/XA.
- **Generated code identical for OS and CMS operating systems.** You can move modules between MVS and CMS without even recompiling.
- **Complete libraries.** We have implemented all the library routines described by Kernighan and Ritchie (the informal C standard), and all the library routines supported by Lattice (except operating system dependent routines), plus extensions for dealing with 370 operating environments directly. Especially significant is our byte-addressable Unix®-style I/O access method.
- **Built-in functions.** Many of the traditional string handling functions are available as built-in functions, generating in-line machine code rather than function calls. Your call to move a string can result in just one MVC instruction rather than a function call and a loop.

In addition to mainframe software development, you can also use our new cross-compiler to develop PC software on your IBM mainframe. With our cross-compiler, you can compile Lattice C programs on your mainframe and generate object code ready to download to your PC.

With the cross-compiler, we also offer PLINK86™ and PLIB86™ by Phoenix Software Associates Ltd. The Phoenix link-editor and library management facility can bind several compiled programs on the mainframe and download immediately executable modules to your PC.

Tomorrow...

We believe that the C language offers the SAS System the path to true portability and maintainability. And we believe that other companies will make similar strategic decisions about C. Already, C is taught in most college computer science curriculums, and is replacing older languages in many. And almost every computer introduced to the market now has a C compiler.

C, the language of choice...

C supports structured programming with superior control features for conditionals, iteration, and case selection. C is good for data structures, with its elegant implementation of structures and pointers. C is conducive to portable coding. It is simple to adjust for the size differences of data elements on different machines.

Continuous support...

At SAS Institute, we support all our products. You license them annually; we support them continuously. You get updates at no additional charge. We have a continuing commitment to make our compiler better and better. We have the ultimate incentive—all our software products depend on it.

For more information...

Complete and mail the coupon today. Because we've got the development tool for your tomorrow.

SAS Institute Inc.
SAS Circle, Box 8000
Cary, NC 27511-8000
Telephone (919) 467-8000 x 7000

I want to learn more about:

- ☐ the C compiler for MVS software developers
- ☐ the C compiler for CMS software developers
- ☐ the cross-compiler with PLINK86 and PLIB86

today...so I'll be ready for tomorrow.

Please complete or attach your business card.

Name ______________________

Title ______________________

Company ______________________

Address ______________________

City ______________ State ________ ZIP ________

Telephone ______________________

Mail to: SAS Institute Inc., Attn: CC, SAS Circle, Box 8000, Cary, NC, USA. 27511-8000. Telephone (919) 467-8000, x 7000

DDJ 12/86

Borland's Turbo Lightning™ FREE

Enlarged *Shift* keys

Two *Ctrl* and *Alt* keys for ambidextrous access.

Isolated *Escape* key.

Positive-tactile, firm-feel keys.

12 *Function* keys for increased automatic operation.

Extra-wide, easy-to-find *Return* key. *IBM changed sizes again by shrinking their Return key.*

Dedicated *Function Control* keys.

Caps, Num and *Scroll* Lock indicator lights. *IBM has no lights on their XT.*

Separate dedicated *Numeric Pad* with enlarged *Enter* key, four *Arithmetic Function* keys.

Separate *Cursor* and *Screen Control* keys.

Switch-selectable compatibility with IBM, PC, XT, AT, PCjr, AT&T and all compatibles. *IBM's new Enhanced Keyboard runs only on their new XT's, AT's and ignores their installed base which probably includes you.*

Switch allows you to swap position of *Ctrl* and *Caps* lock. *IBM moved Ctrl to bottom row, you have no choice!*

Here it is...DATADESK'S TURBO-101 ENHANCED KEYBOARD®

for the 10 million PC users IBM just ignored!

IBM just announced their new redesigned "standard" keyboard for personal computers. There's only one problem: *it won't work* on your IBM computer if it was purchased prior to June 1986 or on *any* PC compatible purchased at any time!

Not to worry. Our new *Turbo-101* Enhanced Keyboard gives you the layout and enhancements of the IBM with some logical improvements (see above photo). And it works on your existing PC, XT, AT, PCjr, AT&T, Epson and virtually all compatibles!

Get Borland's Turbo Lightning ™ For FREE!

To really turbocharge your productivity, we are including, free-of-charge, Borland's red-hot Turbo Lightning software with each keyboard. Now, when using SideKick, WordPerfect, Microsoft Word, 1-2-3 or most popular programs, our Turbo-101 Keyboard will check your spelling *as you type,* gives you instant access to Random House's 80,000-word Concise Dictionary and 60,000-word Thesaurus and much, much more!

"Lightning's good enough to make programmers and users cheer, executives of other software companies weep," says Jim Seymour of PC Week. Sold separately, Turbo Lightning retails for $99.95!

The Turbo-101 is the best data entry tool since the pencil!

For users of spreadsheets like 1-2-3, the Turbo-101's *separate* cursor controls and numeric keypad makes entering numeric data into cells and moving from cell to cell as natural as moving your fingers. And for *word-processing,* the *'Selectric'* typewriter layout makes the Turbo-101 as easy to use as a pencil; and with the extra large Enter, Shift & Control Keys, you'll make so few mistakes, you won't even need an eraser!

SPECIAL OFFER!

ONLY $149.95* *FOR BOTH KEYBOARD & SOFTWARE*

Includes 30-day money back guarantee and 2 year full warranty. To prove that we don't ignore you or your pocketbook, you get our Turbo-101 Enhanced Keyboard and Borland's Turbo Lightning for an astounding $149.95.* No, you didn't read it wrong. During this amazing ***Introductory Offer*** you get both keyboard and software for less than most software programs by themselves! Now, if you're still feeling ignored, you can always do what you-know-who wants you to do...*and buy a new computer to get their keyboard!*

Up to now, DataDesk International may be one of the best kept secrets, but here's what's being said about our first *end-user* Keyboard/Borland software bundle:

"Who Can Pass Up a Deal? Department. Talk about an aggressive product!" John C. Dvorak, InfoWorld Mar 86

"It solves all of the problems exhibited by their regular PC/XT keyboard...it's a great bargain!" PC Productivity Digest May 86

"DataDesk Intl. has designed a sturdy and handsome keyboard that has tactile response...is the hardware bargain of the year" says Charles Humble, Oregonian Jan 86

"The best part of the keyboard is the way it feels. It's ideal! And fast. I've never worked on a keyboard with a nicer touch." Business Computer Digest Aug 86

"It's a good keyboard. Good feel; the keys have tactile feedback. No mush at all. This is about as good a keyboard deal as you're likely to find...I have absolutely no hesitation in recommending the Model PC8700." Jerry Pournelle, Byte Magazine Sept. 86

"This keyboard is neat to type on and feels solid. It has tactile feedback keys...I can type much faster on it." Test Drive Scorecard: DataDesk-10 Key Tronics-9 Teleconnect Magazine May 86

BOTH TURBO-101 ENHANCED™ KEYBOARD AND BORLAND'S TURBO LIGHTNING™ SOFTWARE FOR ONLY:

$149.95* **LIMITED OFFER**

*Plus $10 Shipping & Handling. California Residents Add $9.75 Per Unit Sales Tax

NO. UNITS: ____

PAYMENT: ☐ VISA ☐ MC ☐ CHECK

AMOUNT ENCLOSED: $ ____

Enter

NAME: ____

ADDRESS: ____

CITY: ____ STATE: ____ ZIP: ____

PHONE: ____ COMPUTER TYPE: ____

CC NO.: ____ EXP: ____

The Only Alternative

datadesk™

INTERNATIONAL

7650 Haskell Avenue
Van Nuys, California 91406 (818) 780-1673

credit card orders call

(800) 826-5398

in CA call
(800) 592-9602

Turbo-101 is a trademark of DataDesk International. Turbo Lightning is a trademark of Borland International. IBM and IBM AT are registered trademarks of International Business Machines, Inc.

*Price does not include adaptor cables required by certain compatibles • A Limited offer—price subject to change without notification.

MADE IN USA

Circle **no. 329** on reader service card.
Circle **no. 330** on reader service card.

These Softstrips by Cauzin Systems contain the listing for Richard Rodman's 32000 cross assembler.

1 2 3 4 5 6 7

8 9 10 11 12 13

Softstrip
COMPUTER READABLE PRINT

Listing One (Text begins on page 104.)

```
Listing One
;
; General string comparison routine for 8086
; by Ray Duncan, June 1986
;
; Call with:    DS:SI = address of string1
;               DX    = length of string1
;               ES:DI = address of string2
;               BX    = length of string2
;
; Returns:      Z and S flags set appropriately:
;               Z     = True  if strings are equal
;               or
;               Z     = False if strings are not equal, and
;               S     = True  if string1 < string2
;               S     = False if string1 > string2
;
strcmp  proc    near
        mov     cx,bx                   ; set length to compare
        cmp     dx,bx                   ; use shorter of two lengths
        ja      scmp1
        mov     cx,dx
scmp1:  repz cmpsb                      ; now compare strings
        jz      scmp2                   ; jump, strings equal so far
        ret                             ; return, strings not equal, Z=False
scmp2:  sub     dx,bx                   ; compare original string lengths
        ret                             ; return with S and Z flags set
strcmp  endp
```

End Listing One

Listing Two

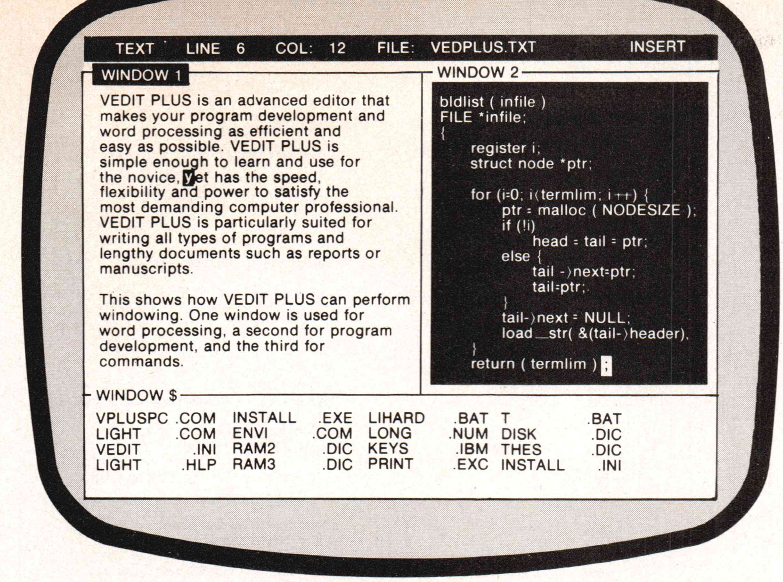
```
Listing Two
;
; General string comparison routine for 68000
; by Rick Wilton, June 1986
;
; Call with:    A0    = address of string1
;               D0    = length of string1
;               A1    = address of string2
;               D1    = length of string2
;
; Returns:      D3    = flag (-1,0,1)
;                       -1 if string1 < string2
;                        0 if string1 = string2
;                        1 if string1 > string2
;
strcmp                                  ; set d2 = shorter length
        move.b  d1,d2                   ; d2 := length2
        cmp.b   d0,d1
        blt.s   strcmp1                 ; branch if length2 < length1
        move.b  d0,d2                   ; d2 := length1

strcmp1 subq.w  #1,d2
        bmi.s   strcmp3                 ; branch if string length=0

strcmp2 cmpm.b  (a0)+,(a1)+             ; compare strings
        dbne    d2,strcmp2
        bne.s   strcmp4                 ; branch if strings unequal

strcmp3 cmp.b   d0,d1
        bne.s   strcmp4                 ; branch if lengths unequal
        moveq   #0,d3                   ; string1 = string2, return 0
        rts

strcmp4 bmi.s   strcmp5                 ; branch if d1 < d0
        moveq   #-1,d3                  ; string1 < string2, return -1
        rts

strcmp5 moveq   #1,d3                   ; string1 > string2, return 1
        rts
```

End Listing Two

Listing Three

```
Listing Three
Title EXEC program using undocumented MS-DOS Interrupt 2EH

;
; This little demonstration program illustrates the undocumented
; MS-DOS Int 2EH route to the command interpreter in COMMAND.COM.
; In this example we just pass the command tail of the line that
; loaded the EXEC2E program.
;
; by David Gwillim - 20 June 1986
; Appears to work on all current versions of MS-DOS (2.x-3.1)
; Adapted from Turbo Pascal program written by Russ Nelson, Potsdam, NY.
;

cseg    segment

        org  100H

        assume cs:cseg,ds:nothing       ; force assembler to provide
                                        ; CS overrides in the right places

begin:  jmp     start

        db      20 dup ('STACK ')
stack   equ     $

stkseg  dw      0                       ; save SS register
stkptr  dw      0                       ; save SP register

msg1    db      'Beginning DOS Int 2Eh Exec',13,10,'$'
```


(continued on page 88)



VEDIT PLUS

#1 CHOICE IN PROGRAMMABLE EDITORS

VEDIT PLUS has been the #1 choice of professionals since 1980. Our latest release is even better - you can open windows to simultaneously edit several files, access many editing functions with pop-up menus, use keystroke macros to speed editing, and run other programs or DOS commands from within VEDIT PLUS.

Whether your needs are program development, technical writing or word processing, VEDIT PLUS is your answer. With over 40,000 users you can depend on VEDIT PLUS to perform consistently and reliably. It is simple enough to learn for the novice, yet has the speed, flexibility and power to satisfy the most demanding professional.

Compare. No other editor is as powerful - unlimited keystroke macros, instant 'off-the-cuff' command macros utilizing a complete programming language, single command file comparison, special word processing and programming features. No other editor is as easy to use - on-line help, pop-up menus, 75 page tutorial, 380 page manual, and VEDIT PLUS is completely customizable.

Fully supports color windows on IBM CGA & EGA, and even windows on most CRT terminals (including CRT's connected to an IBM PC). Available for IBM PC, TI Professional, Tandy 2000, DEC Rainbow, Wang PC, MS-DOS, CP/M-86 and CP/M-80. Order direct or from your dealer. $185

> "To sum things up, VEDIT PLUS is a small, fast, sophisticated editor with a wealth of features and a good macro language. It offers many rewards for the dedicated programmer."
> **Computer Language, Chris Wolf, Scott Lewis, Mark Gayman 6/86**

> "VEDIT PLUS is a wholly remarkable program: blindingly fast, extremely powerful, and highly flexible."
> **Profiles Magazine, Robert Lavenda 4/86**

VEDIT PLUS FEATURES

- Simultaneously edit up to 37 files of unlimited size.
- Split the screen into variable sized windows.
- 'Virtual' disk buffering simplifies editing of large files.
- Memory management supports up to 640K.
- Execute DOS commands or other programs.
- MS-DOS pathname and CP/M user number support.
- Horizontal scrolling - edit long lines.
- Flexible 'cut and paste' with 36 text registers.
- Customization - determine your own keyboard layout, create your own editing functions, support any screen size, any CRT.
- Optimized for IBM PC/XT/AT. Also 132 column & up to 70 lines.

EASY TO USE

- Interactive on-line help is user changeable and expandable.
- On-line integer calculator (also algebraic expressions).
- Single key search and global or selective replace.
- Pop-up menus for easy access to many editing functions.
- Keystroke macros speed editing, 'hot keys' for menu functions.

FOR PROGRAMMERS

- Automatic Indent/Undent for 'C', PL/I or PASCAL.
- Match/check nested parentheses, i.e. '{' and '}' for 'C'.
- Automatic conversion to upper case for assembly language labels, opcodes, operands with comments unchanged.
- Optional 8080 to 8086 source code translator.

FOR WRITERS

- Word Wrap and paragraph formatting at adjustable margins.
- Right margin justification.
- Support foreign, graphic and special characters.
- Convert WordStar and mainframe files.
- Print any portion of file; separate printer margins.

MACRO PROGRAMMING LANGUAGE

- 'If-then-else', looping, testing, branching, user prompts keyboard input, 17 bit algebraic expressions, variables.
- CRT emulation within windows, Forms entry.
- Simplifies complex text processing, formatting, conversions and translations.
- Complete TECO capability.
- Free macros: • Full screen file compare/merge • Sort mailing lists • Print Formatter • Main menu

Circle **no. 122** on reader service card.

CompuView

1955 Pauline Blvd., Ann Arbor, MI 48103 (313) 996-1299, TELEX 701821

Listing Three

(Listing continued, text begins on page 104.)

```
msg2    db      'Terminating DOS Int 2Eh Exec',13,10,'$'

start:  push    cs                      ;make our local data addressable
        pop     ds

        mov     dx,offset msg1          ;display sign-on message
        mov     ah,9
        int     21h

        mov     sp,offset stack         ;reset SP to our own internal stack

        mov     bx,offset cs:endcode    ;Get the offset of the end of our code
        shr     bx,1                    ;divide by 16 to get paragraphs
        shr     bx,1
        shr     bx,1
        shr     bx,1
        inc     bx                      ;round paragraphs up
        mov     ah,4ah                  ;shrink down this COM program's
        int     21h                     ;memory allocation to what's needed

        mov     stkseg,ss               ;save our current SS reg
        mov     stkptr,sp               ; and SP reg values

        mov     si,80h                  ;let DS:SI point to the command
                                        ;to be executed

        int     2eh                     ;undocumented DOS exec interrupt
                                        ; any error code is now in AX

        mov     ss,stkseg               ;Restore SS and SP registers
        mov     sp,stkptr

        push    cs                      ;restore local addressing
        pop     ds

        mov     dx,offset msg2          ;say we're done
        mov     ah,9
        int     21h

        mov     ax,4c00h                ;exit to DOS
        int     21h

endcode equ     $

cseg    ends

        end     begin
```

End Listing Three

Listing Four

```
Listing Four

/*
        The following little program can set an environment variable
        whose name is the first command line argument, and whose new
        value is the second command line argument.  For example, to
        set environment variable XYZ to a value of HELLO, you would
        run this program using the command SETVAR XYZ HELLO.

        The *PURPOSE* of this piece of code is to illustrate an un-
        documented DOS interrupt entry point (2Eh) that will execute
        *ANY* DOS command *WITHOUT* having to load another copy of
        COMMAND.COM.  Not only is it faster, but it is the ONLY way
        to set an environment variable (short of peeking and poking
        around into memory).  You *CAN'T* set an environment variable
        with the command exec("COMMAND.COM", "/CSETXYZ=HELLO") because
        when the second copy of COMMAND.COM is loaded, it gets its
        very own environment (a duplicate of the parent's).  Although
        the SET command WILL modify that duplicate copy, it won't
        modify the parent's!

        When I said that interrupt 2Eh can be used to execute *ANY*
        DOS command, I meant just that!  You can leave off the filename
        extension (as you normally do at the command line), and it will
        perform the normal search for COM, EXE, and BAT files to execute,
        or even execute built-in commands as we've seen above.

        Enjoy!

        Dan Lewis, owner
        Key Software Products
        440 Ninth Avenue
        Menlo Park, CA 94025
        (415) 364-9847

        P.S.- This *IS* an undocumented "feature" of DOS 2.xx.  I have
        NO idea if 3.xx etc support it!

*/

#include <stdio.h>

main(argc, argv)
int argc ;
char *argv[] ;
        {
        Set_Var(argv[1], argv[2]) ;
        }


Set_Var(variable, value)
char *variable, *value ;
        {
        char setbfr[100] ;
```


(continued on page 90)

Listing Four

(Listing continued, text begins on page 104.)

```
        /* Build the command line: "SET <var>=<value>" */

        strcpy(setbfr, "SET ") ;
        strcat(setbfr, variable) ;
        strcat(setbfr, "=") ;
        strcat(setbfr, value) ;

        /* Now use INT 2Eh to execute it! */

        Execute_String(setbfr) ;
        }

       /* ---------------------------------------------- */
       /* The really interesting stuff starts here.... */
       /* ---------------------------------------------- */


Execute_String(s)
char *s;
        {
        long vec22, vec23, vec24 ;
        long Get_Vec(vector) ;
        static char bfr[81];

        /* Concatenate a carriage return onto end of command line */

        strcpy(bfr + 1,s) ;
        *bfr = strlen(bfr + 1) ;
        strcat(bfr + 1,"\r") ;

        /* preserve cntrl-break, terminate, and critical error vectors */
        vec22 = Get_Vec(0x22) ;
        vec23 = Get_Vec(0x23) ;
        vec24 = Get_Vec(0x24) ;

        Release_Memory() ;        /* necessary! */
        Exec(bfr) ;               /* execute command */

        /* reset cntrl-break, terminate, and critical error vectors */
        Set_Vec(0x22,vec22) ;
        Set_Vec(0x23,vec23) ;
        Set_Vec(0x24,vec24) ;
        }


Exec(s)
char *s ;
        {
#asm
        ; preserve ds, bp, ss, & sp

        push    ds
        push    bp
        mov     cs:WORD save_ss,ss
        mov     cs:WORD save_sp,sp

        ; ds:si ->CR/NULL-terminated command line string

        mov     si,[bp+4]
        int     2Eh

        ; restore preserved registers
        mov     ss,cs:WORD save_ss
        mov     sp,cs:WORD save_sp
        pop     bp
        pop     ds

        jmp     Rtn

save_ss:dw      0
save_sp:dw      0
save_ds:dw      0
save_bp:dw      0

Rtn:
#end
        }


Release_Memory()
        {
        if (Release())
                {
                puts("Release Memory Failure\n") ;
                exit(1) ;
                }
        }


Release()
        {
#asm
        mov     ax,cs
        sub     ax,0010h
        mov     es,ax
        mov     bx,ds
        add     bx,1000h
        sub     bx,ax
        mov     ah,4Ah
        int     21h
        mov     ax,0
        jnc     R_Rtn
        inc     ax
R_Rtn:
#end
        }

long Get_Vec(vector)    /* Uses DOS function 35h to fetch an interrupt vctr */
unsigned vector ;
        {
```

```
#asm
        mov     ah,35h
        mov     al,BYTE [bp+4]
        int     21h
        mov     ax,bx
        mov     dx,es
#end
        }

Set_Vec(vector,addr)    /* Uses DOS function 25h to set an interrupt vctr */
unsigned vector ;
long addr ;
        {
#asm
        mov     ah,25h
        mov     al,BYTE [bp+4]
        mov     bx,WORD [bp+6]
        mov     es,WORD [bp+8]
        int     21h
#end
        }
```

End Listing Four

Listing Five

```
Listing Five
/*
 *      SYSDEF.H
 *      Standard system definitions
 *      Ross P. Nelson
 */

/* Expanded public/external syntax */
#define FORWARD extern
#define IMPORT      extern
#define PUBLIC

#ifdef DEBUG        /* Generate symbols if debugging, else don't export */
#define LOCAL
#else
#define LOCAL       static
#endif
/* System constants and data types */
#define TRUE        1
#define FALSE       0

typedef unsigned char       byte;
typedef byte                boolean;
typedef unsigned short      word;
typedef unsigned short      selector;


/*
 *      STD.C    Copyright (C) 1985 Ross P. Nelson
 *
 *          Redirect stdin/stdout.  Usually called before performing an
 *          EXEC, so that the child task will read or write via a file.
 *          Sample usage is in the MAIN routine below.
 */

#include <stdio.h>
#include <dos.h>
#include <iosl.h>
#include <sysdef.h>

IMPORT  word        _PSP[2]; /* offset, segment */
IMPORT  struct UFB _ufbs[];

/*
 *      SETSTDIO
 *          Set stdin or stdout to the file.  Caller is responisble for
 *          having opened the file in the corrent mode and positioning it
 *          as necessary.  Returns a value to be used when reseting stdio
 *          to original value.
 */
PUBLIC int setstdio (stdfp, newfp)
FILE *stdfp, *newfp;
{
byte reset, save;
int handle, redir;

        handle = _ufbs[fileno (newfp)].ufbfh;
        redir = _ufbs[fileno (stdfp)].ufbfh;
        save = 0xFF;
        peek (_PSP[1], 0x18 + handle, &reset, 1);
        poke (_PSP[1], 0x18 + handle, &save, 1);
        peek (_PSP[1], 0x18 + redir, &save, 1);
        poke (_PSP[1], 0x18 + redir, &reset, 1);
        return (int) save;
}

/*
 *      RESTDIO
 *          Must be called to reset stdio values to original.  Caller is
 *          responsible for closing file after restdio.
 */
PUBLIC void restdio (stdfp, newfp, reset)
FILE *stdfp, *newfp;
int reset;
{
byte direct, old;
int handle, redir;

        handle = _ufbs[fileno (newfp)].ufbfh;
        redir = _ufbs[fileno (stdfp)].ufbfh;
        old = (byte) reset;
        peek (_PSP[1], 0x18 + redir, &direct, 1);
        poke (_PSP[1], 0x18 + redir, &old, 1);
        poke (_PSP[1], 0x18 + handle, &direct, 1);
}

#if (PROTOTYPE)
abort (s)
char *s;
{
        fprintf (stderr, s);
        exit (4);
```

(continued on next page)

16-BIT

Listing Five

(Listing continued, text begins on page 104.)

```
}

main ()
{
FILE *f;
int reset;

        puts ("Begin test - opening file TEST.XX");
        f = fopen ("test.xx", "w");
        if (f == NULL)
                abort ("open failed");
        reset = setstdio (stdout, f);                   /* stdout <- f */
        fprintf (stderr, "invoking LS\n");      /* can't write to stdout */
        if (forklp ("ls.exe", "ls", NULL)) {    /* run LS */
                restdio (stdout, f, reset);  /* clean up if cant exec */
                abort ("exec failed");
                }
        (void) wait ();                         /* wait til LS done */
        restdio (stdout, f, reset);             /* stdout <- original value */
        fclose (f);
        puts ("Test completed - LS results in TEST.XX");
}
#endif
```

End Listing Five

Listing Six

```
Listing 6 for November 86 16-Bit Column in DDJ
----------------------------------------------

;***************************************************************
;*                                                             *
;* Wildcard filename expansion for MS-DOS 2.00 and later.      *
;*                                                             *
;* By: Randy Langer, MicroSphere Technology                    *
;*                                                             *
;***************************************************************

ifndef  model                           ; if default model (both small)

model   equ     0                       ; 0 = small code, small data
                                        ; 1 = large code, small data
                                        ; 2 = small code, large data
endif                                   ; 3 = large code, large data

codeseg segment byte public 'code'
        assume  cs:codeseg

        public  wildcard_

if      model and 1

wildcard_       proc    far

else

wildcard_       proc    near

endif

        push    bp                      ; save frame pointer
        mov     bp,sp                   ; point to our stack

if      model and 2

        push    ds                      ; save DS if large data
        mov     ds,[bp+6]               ; and get segment of struct ptr

endif

        mov     bx,[bp+4]               ; get offset of struct ptr
        mov     al,[bx]                 ; get flag byte
        or      al,al                   ; see if high bit set
        js      rtn_null                ; if so, no more to find
        push    es                      ; save reg used by DOS call
        mov     ah,47                   ; get current DTA addr
        int     21h                     ; do it
        mov     ax,es                   ; save DTA segment
        pop     es                      ; restore ES
        push    ds                      ; save for later restoration
        push    ax                      ; save addr of old DTA
        push    bx
        mov     dx,[bp+4]               ; get ptr to user's struct
        add     dx,2                    ; point past flag bytes
        mov     ah,26                   ; set "new" DTA addr
        int     21h
        mov     bx,[bp+4]               ; get entry pointer again
        mov     cl,[bx+1]               ; set search attributes
        mov     ah,79                   ; set token for search next
        test    byte ptr [bx],1         ; if this is really search next
        jnz     not_1st                 ;  branch
        inc     byte ptr [bx]           ; else, set flag
        dec     ah                      ; and set token for search first
not_1st:
        mov     dx,[bx+45]              ; get offset to filespec

if      model and 2

        mov     ds,[bx+47]

endif

        int     21h                     ; do the search
        pop     dx                      ; get addr of old DTA
        pop     ds
```

(continued on page 94)

CAD — THE BEST IS EVEN BETTER!

ProDesign II — The Easy-to-Use CAD System. It's the NUMBER ONE CAD PACKAGE for under $1000. And it costs only $299. With over 25,000 satisfied users, ProDesign II outshines anything in its class. It outshines many packages costing $2000 and more! ProDesign II offers more features, more compatibility, and superior ease-of-use.

But now, THE BEST IS EVEN BETTER!

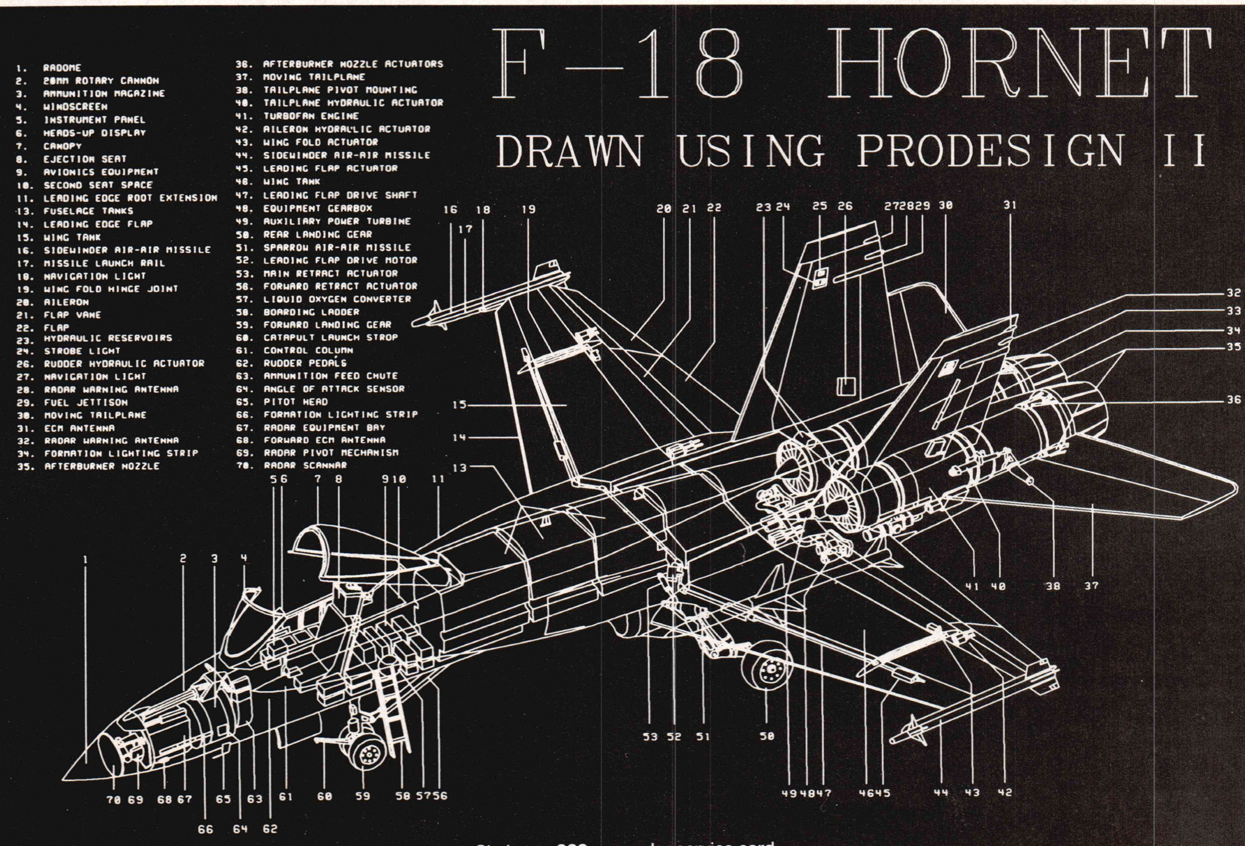
Version 2.5 of ProDesign II is now available. More than 50 additional features have been added to ProDesign II. Features such as Pattern Hatching, On-Screen Menus, Geometric Calculations, Macros, Parallel Lines, Parallel Curves, enhanced Auto-Dimensioning, Block Arrays and Circular Arrays, to name a few. Features that make ProDesign II even more powerful. Features that make ProDesign II even easier to use.

How much more does version 2.5 cost? None! The new ProDesign II is still $299! Where can you get it? See your local dealer, or contact us:

AMERICAN SMALL BUSINESS COMPUTERS
118 SOUTH MILL STREET
PRYOR, OK 74361
(918) 825-4844

Telex 9102400302

PRODESIGN II — ONLY $299!



Listing Six (Listing continued, text begins on page 104.)

```
         push    ax              ; save return code from search
         mov     ah,26           ; restore DTA ptr
         int     21h
         pop     ax              ; restore return code
         pop     ds              ; get back segment of user struct
         mov     bx,[bp+4]       ; and its offest
         or      ax,ax           ; see if search successful
         jz      rtn_name        ; branch if so
         mov     byte ptr [bx],128 ; else, say no more
rtn_null:
         xor     ax,ax           ; return null ptr
         mov     dx,ax           ; in case of large data
wild_end:

if       model and 2

         pop     ds

endif

         pop     bp              ; restore frame pointer
         ret                     ; and return
rtn_name:
         mov     dx,ds           ; in case of large data model
         mov     ax,32           ; offset to file name
         add     ax,bx           ; add to struct base
         jmp     wild_end        ; and exit

wildcard_        endp
codeseg ends
         end
```

End Listing Six

Listing Seven

```
Listing 7 for November 86 16-Bit Column in DDJ:
-----------------------------------------------

/*

      WILDCARD.H  for use with  WILDCARD.ASM
      by Randy Langer, Microsphere Technology

*/

typedef struct
          {
                    char    flag;
                    char    att_sel;
                    char    tempdata[21];           /* don't mess
                                                       with this */
                    char    f_atts;
                    long    datetime;
                    long    filesize;
                    char    filename[13];
                    char    *filespec;

          } W_CARD;

char      *wildcard();
```

End Listing Seven

Listing Eight

```
Listing 8 for November 1986 16-Bit Column in DDJ
------------------------------------------------

/*
 *  WILDTEST.C  Program to demonstrate use of WILDCARD.ASM
 *  by Randy Langer, MicroSphere Technology.
 *
 */

#include        "stdio.h"
#include        "wildcard.h"

main(argc, argv)

int             argc;
STR             argv[];
{

W_CARD  y;
STR     s;

          while(--argc)
          {
                    y.filespec = *++argv;
                    y.att_sel = 0x10;
                    y.flag = 0;
                    while(s = wildcard(&y))
                              printf("%s\n", s);
          }
}
```

End Listings

GET A HEAD START FOR 1987

With these SUPERior Programmer's Packages

The **A**spen **S**ystems **S**ubroutine **E**ditor (**ASE**) you can call from your programs. Design your own screen layouts - - update in several windows simultaneously. **ASE** is:

- ★ **TRANSPORTABLE** - can be configured for any keyboard and almost any computer or language under MSDOS.
- ★ **VERSATILE** - customize input for any application.
- ★ **COMPLETE** - with full screen edit capabilities and a wide variety of automatic conversions.
- ★ **EASY TO USE** - data and screen layouts described in a single map.
- ★ **FULLY DOCUMENTED** - including examples in BASIC, PASCAL, FORTRAN, and even COBOL.
- ★ **AFFORDABLE** -
  - **ASE** is still only............................. **$99**
  - **DEMO** (See what ASE can do for you!)......... **$ 3**

The **A**spen **S**ystems **S**ubroutine **P**ackage (**ASP**) - with many functions unavailable or difficult to perform in high level languages. **ASP** is:

- ★ **TRANSPORTABLE** - can be configured for almost any computer or language under MSDOS or CP/M.
- ★ **FULLY DOCUMENTED** - a 300 page manual, with examples in BASIC, PASCAL, FORTRAN, and even COBOL.
- ★ **EFFICIENT** - faster, with smaller memory requirements than some high level languages.
- ★ **AFFORDABLE** - **ASP** is still only................ **$130**

ASP includes 150+ subroutines: conversions; sorts; string, bit, date/time functions; decimal arithmetic, and **MORE**.

Prices are PPD (continental USA).
Colorado Residents add 3%.

P. O. Box 1163
Grand Junction, CO 81502
(303) 245-3262
VISA/MasterCard accepted

Aspen Systems

CP/M and MS-DOS are trademarks of Digital Research and Microsoft, respectively.

Circle **no. 277** on reader service card.

ATTENTION

C·PROGRAMMERS

File System Utility Libraries

All products are written entirely in K&R C. Source code included, No Royalties, Powerful & Portable.

BTree Library **75.00**

- High speed **random** and **sequential** access.
- **Multiple keys** per data file with up to **16 million** records per file.
- **Duplicate keys,** variable length data records.

ISAM Driver **40.00**

- Greatly speeds application development.
- Combines ease of use of **database manager** with flexibility of **programming language.**
- Supports **multi key files** and **dynamic index definition.**
- Very **easy to use.**

Make **59.00**

- Patterned after the **UNIX** utility.
- Works for programs written in **every language.**
- **Full macros, File name expansion** and **built in rules.**

Full Documentation and Example Programs Included.

ALL THREE PRODUCTS FOR — 149.00

For more information call or write:

softfocus

1343 Stanbury Drive
Oakville, Ontario, Canada
L6L 2J5
(416) 825-0903

Credit cards accepted.

Dealer inquiries invited.

Circle **no. 259** on reader service card.

SMILES FOR CHRISTMAS

Only the thrill of giving can rival the excitement and appreciation your favorite programmer will feel while opening your thoughtful gift of JACK™, The Resident Program Developers Kit.

JACK contains everything required to easily create RAM resident software without the problems of DOS re-entrancy, memory management or windows.

JACK FEATURES:

Virtually any EXISTING C or ASSEMBLER program can be made memory resident without any modification to the existing source code.

Screens are automatically saved when popping up and restored upon popping down.

Your choice of Hotkey and interrupt vector for program use.

You can issue DOS function calls and perform disk I/O in your memory resident program.

No more system crashes and lost data. All programs developed with JACK will peacefully coexist.

JACK applications can be loaded and popped-up in any order.

Programs developed with JACK will not become obsolete because JACK does not make any use of undocumented DOS calls. [JACK requires DOS Version 2.0 or higher]

And just in time for Christmas, CrackerJack would like to introduce BEYOND, a library of memory management and service routines for the Intel Above Board. Now all of your programs can take advantage of expanded memory. With our library and Intel's ABOVE Board, your programs can go ABOVE and BEYOND their previous limitations.

To Order, please send $199.95 [US] + S/H for JACK or $49.95 [US] + S/H for BEYOND by certified check or international money order to:

Crackerjack Microsoftware Corporation.
200 Bay Street PO Box 86,
Toronto, Ontario M5J-2J2
Canada.

(416) 865-9621.

Circle **no. 238** on reader service card.

When ordering, please specify either the assembler version or C version of JACK and indicate the vendor and release level of your compiler.

Circle **no. 238** on reader service card.

Listing One (Text begins on page 108.)

```
Listing 1.  Contents of Turbo Pascal included ProcParm.INC file.

{                              ProcParm.INC           Version 1.1    86/05/07

 See ProcParm.PAS for an explanation.

 Author: Mike Babulic     Compuserve ID: 72307,314     FIDO: 134/1
         3827 Charleswood Dr. N.W.
         Calgary, Alberta,
         CANADA
         T2L 2C7

}

procedure Call_ProcParm;
  begin
  Inline
    ($89/$EC/          {      MOV   SP,BP          ;Drop down one level }
     $5D/              {      POP   BP             }
     $8B/$66/$02/      {   SS:MOV   SP,[BP+2]  ;Exchange Return Addr &
                                                    Procedure Ptr}
     $87/$66/$04/      {   SS:XCHG SP,[BP+4] }
     $89/$66/$02       {   SS:MOV   [BP+2],SP }
    )
  end;
```

End Listing One

Listing Two

```
Listing 2.  Contents of file ProcPar.QK.

{                           ProcParm.QK            Version 1.0    86/04/22


 Author: Mike Babulic     Compuserve ID: 72307,314     FIDO: 134/1
         3827 Charleswood Dr. N.W.
         Calgary, Alberta,
         CANADA
         T2L 2C7
}
  Inline(
     $8B/$66/$02/      {   SS:MOV   SP,[BP+2]       ;Exchange Return Addr &
                                                      Procedure Ptr}

     $87/$66/$04/      {   SS:XCHG SP,[BP+4] }
     $89/$66/$02/      {   SS:MOV   [BP+2],SP }
     $89/$EC/          {      MOV   SP,BP           ;Standard Turbo Return
                                                  (if no Parameters)}

     $5D/              {      POP   BP              }
     $C3               {      RET                   ;Near Return }
     )
```

End Listing Two

Listing Three

```
Listing 3.  Turbo Pascal demo program for procedural parameters.

program proc_param_demo;

CONST FIRST = 1;
      LAST  = 1000;


TYPE Vector =  ARRAY [FIRST..LAST] OF INTEGER;

VAR A : Vector;
    I, Start, Finish : INTEGER;

(*-------------------------------------------- Shell_Sort -------*)


PROCEDURE Shell_Sort(VAR A : Vector);

VAR I, J, Offset, Skip, Tempo, NData : INTEGER;
    In_Order : BOOLEAN;

BEGIN
    NDATA := LAST - FIRST + 1;
    Skip := NDATA;
    WHILE Skip > 1 DO BEGIN
        Skip := Skip DIV 2;
        REPEAT
            In_Order := TRUE;
            FOR J := FIRST TO LAST - Skip DO BEGIN
                I := J + Skip;
                IF A[J] > A[I] THEN  BEGIN
                    In_Order := FALSE;
                    Tempo := A[I];
                    A[I]  := A[J];
                    A[J]  := Tempo
                END; (* IF *)
            END;  (* FOR *)
        UNTIL In_Order;
    END;  (* WHILE *)
END;  (* Shell_Sort *)

(*-------------------------------------------- QuickSort -------*)

PROCEDURE QuickSort(VAR A : Vector);

    PROCEDURE Sort(Left, Right : INTEGER);

    VAR I, J,
        Pivot, Tempo : INTEGER;

    BEGIN
        I := Left; J := Right;
        Pivot := A[(Left + Right) DIV 2];
        REPEAT
            WHILE A[I] < Pivot DO I := I + 1;
            WHILE Pivot < A[J] DO J := J - 1;
            IF I <= J THEN BEGIN

                Tempo := A[I];
                A[I] := A[J];
                A[J] := Tempo;
                I := I + 1;
                J := J - 1
            END; (* IF *)
        UNTIL I > J;
        IF Left < J THEN Sort(Left,J);
        IF I < Right THEN Sort(I,Right);
    END; (* Sort *)

BEGIN
    Sort(FIRST, LAST)
END; (* QuickSort *)

(*----------------- Use the ProcParm Procedure -----------------*)

{$I PROCPARM.INC}

PROCEDURE Dummy1(VAR A : Vector; P : INTEGER);

BEGIN
   Call_ProcParm;
END; (* Dummy1 *)


PROCEDURE Sort1(VAR A : Vector; P : INTEGER);

BEGIN
    Dummy1(A,P);
END; (* Sort1 *)


(*------------------------ Use Procparm.qk ---------------------*)

PROCEDURE Dummy2(VAR A : Vector; P : INTEGER);

BEGIN
    {$I PROCPARM.QK}
END; (* Dummy2 *)


PROCEDURE Sort2(VAR A : Vector; P : INTEGER);

BEGIN
    Dummy2(A, P)
END; (* Sort2 *)

(*-------------------------------------------- Create_Array -------*)

PROCEDURE Create_Array(VAR A : Vector; Start, Finish : INTEGER);
(* Create a reverse sorted array *)

VAR I : INTEGER;

BEGIN
   FOR I := Start TO Finish DO
       A[I] := Finish + 1 - I
END; (* Create_Array *)


(*-------------------------------------------- Display_Array -------*)

PROCEDURE Display_Array(VAR A : Vector; Start, Finish : INTEGER);

VAR I : INTEGER;
    Dummy : CHAR;

BEGIN
    WRITE('Press <CR> to view array members '); READLN(Dummy); WRITELN;
    FOR I := Start TO Finish DO
        WRITE(A[I]:8);

    WRITELN; WRITELN;
END; (* Display_Array *)


(*------------------------------------------------ Show_Time -------*)
PROCEDURE Show_Time;
(* Procedure to dislplay time *)
TYPE REGTYPE = record
                AX,BX,CX,DX,BP,
                DI,SI,DS,ED,FLAGS  : INTEGER
              END;

     TIME_REC = RECORD
                HOUR, MIN, SEC, HSEC : BYTE
              END;


VAR REGISTER  : REGTYPE;
    AH        : BYTE;
    TIME      : TIME_REC;

BEGIN

  AH   :=  $2C;

     WITH REGISTER, TIME DO BEGIN
        AX:= AH SHL 8;
        MSDOS(REGISTER);
        HOUR :=  Hi(CX);
        MIN  :=  Lo(CX);
        SEC  :=  Hi(DX);
        HSEC :=  Lo(DX);
        WRITELN(' at  ',HOUR,' : ',MIN,' : ',SEC,'.',HSEC);
     END;
END; (* Show_Time *)


BEGIN
   ClrScr;
   WRITELN('Array has index range of ',FIRST,' to ',LAST);
   WRITE('Enter index of first element to view '); READLN(Start); WRITELN;
   WRITE('Enter index of last  element to view '); READLN(Finish); WRITELN;
   IF Start < FIRST THEN Start := FIRST;
   IF (Finish > LAST) THEN Finish := LAST;
   IF Finish < Start THEN Finish := Start + (LAST - FIRST + 1) DIV 10;
```

Techical Product Information.

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

Technical Product Information.

BUSINESS REPLY MAIL

FIRST CLASS PERMIT #217, CLINTON, IOWA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of
Software Tools
FOR THE PROFESSIONAL PROGRAMMER

P.O. Box 2157
Clinton, Iowa 52735-2157

TAKE THIS CARD WITH YOU AS YOU READ THROUGH THIS ISSUE OF DR. DOBB'S.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150
151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170
171	172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189	190
191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210
211	212	213	214	215	216	217	218	219	220
221	222	223	224	225	226	227	228	229	230
231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250
251	252	253	254	255	256	257	258	259	260
261	262	263	264	265	266	267	268	269	270
271	272	273	274	275	276	277	278	279	280
281	282	283	284	285	286	287	288	289	290
291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310
311	312	313	314	315	316	317	318	319	320
321	322	323	324	325	326	327	328	329	330
331	332	333	334	335	336	337	338	339	340
341	342	343	344	345	346	347	348	349	350
351	352	353	354	355	356	357	358	359	360
361	362	363	364	365	366	367	368	369	370
371	372	373	374	375	376	377	378	379	380
381	382	383	384	385	386	387	388	389	390
391	392	393	394	395	396	397	398	399	999

Circle 999 to start a 12 month subscription at the price of $29.97

Dr. Dobb's Journal of Software Tools

Name ____________________

Title ____________________

Company ____________ Phone ____________

Address ____________________

City/State/Zip ____________________

December 1986 #122 Expiration Date: March 31, 1987

Please circle one letter in each category:

I. My work is performed:
A. for in-house use only.
B. for other companies.
C. for end users/retailers.
D. in none of the above areas.

II. My primary job function:
A. Software Project Mgmt/Spvr
B. Hardware Project Mgmt/Spvr
C. Computer Consultant
D. Corporate Consultant
E. Other

III. My company department performs:
A. software development.
B. computer system integration.
C. computer manufacturing.
D. computer consulting.
E. computer research
F. none of the above.

IV. This inquiry is for:
A. a purchase within 1 month.
B. a purchase within 1 to 6 months.
C. product information only.

V. Corporate Purchase Authority:
A. Final Decision-maker
B. Approve/Recommend
C. No Influence

VI. Personal Computer Users at my Jobsite:
A. 10,000 or more
B. 500 to 9,999
C. 100 to 499
D. 10 to 99
E. less than 10

VII. On average, I advise others about computers:
A. more than once per day.
B. once per day.
C. once per week.
D. less than once per week.

VIII. In my job function, I:
A. design software and/or write code.
B. design software.
C. write code.
D. don't design software or write code.

A Reader Service number appears on each advertisement. Circle the corresponding numbers at left for more info.

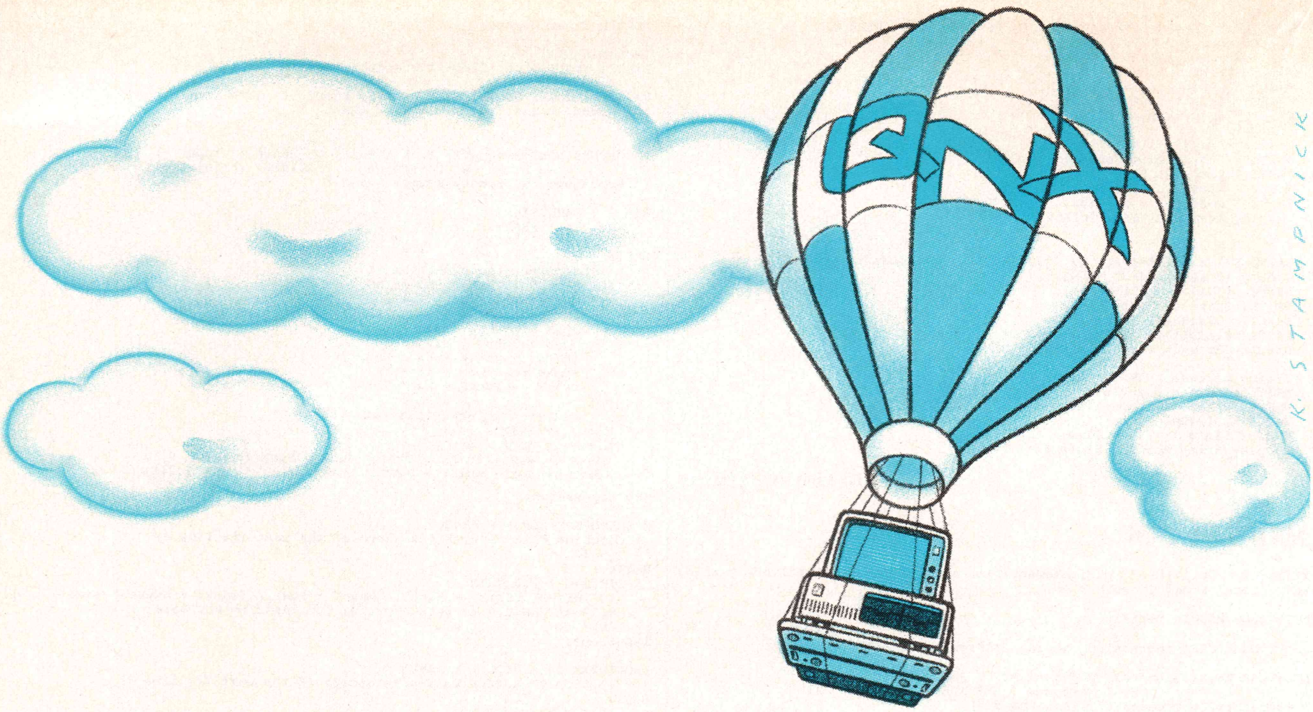
```
   WRITELN('Using ProcParm Procedure '); WRITELN; WRITELN;
   Create_Array(A, FIRST, LAST);
   WRITELN('Using Shell Sort');
   WRITE('Start '); Show_Time;
   Sort1(A,Ofs(Shell_Sort));
   WRITE('Finish'); Show_Time;
   Display_Array(A,Start,Finish);

   Create_Array(A, FIRST, LAST);
   WRITELN('Using QuickSort');
   WRITE('Start '); Show_Time;
   Sort1(A,Ofs(QuickSort));
   WRITE('Finish'); Show_Time;
   Display_Array(A,Start,Finish);


   WRITELN('Using ProcParm.OK  '); WRITELN; WRITELN;
   Create_Array(A, FIRST, LAST);
   WRITELN('Using Shell Sort');
   WRITE('Start '); Show_Time;
   Sort2(A,Ofs(Shell_Sort));
   WRITE('Finish'); Show_Time;
   Display_Array(A,Start,Finish);

   Create_Array(A, FIRST, LAST);
   WRITELN('Using QuickSort');
   WRITE('Start '); Show_Time;
   Sort2(A,Ofs(QuickSort));
   WRITE('Finish'); Show_Time;
   Display_Array(A,Start,Finish);

END.
```

End Listing Three

Listing Four

```
Listing 4.  Definition and implementation modules for BestFit library which
uses a local model InnerWorking.

DEFINITION MODULE BestFit;

EXPORT QUALIFIED Regression, Slope, Intercept, R2;

PROCEDURE Regression(VAR X, Y : ARRAY OF REAL; (* input *)
                     N, LowerBound : CARDINAL  (* input *));
(* Procedure to process arrays X and Y *)

PROCEDURE Slope() : REAL;
(* Function that returns the slope of the best fit line *)

PROCEDURE Intercept() : REAL;
(* Function that returns the intercept of the best fit line *)

PROCEDURE R2() : REAL;
(* Function that returns the goodness of the best fit line *)

END BestFit.


IMPLEMENTATION MODULE BestFit;

FROM MathLib0 IMPORT sqrt;

MODULE InnerWorking;

IMPORT sqrt;
EXPORT Regression, Slope, Intercept, R2;

VAR Sum, SumX, SumXX, SumY, SumYY, SumXY, (* Stat summation *)
    MeanX, MeanY, SdevX, SdevY : REAL;


PROCEDURE Regression(VAR X, Y : ARRAY OF REAL; (* input *)
                     N, LowerBound : CARDINAL  (* input *));
(* Procedure to process arrays X and Y *)

VAR i : CARDINAL;
    Xs, Ys : REAL;

BEGIN
    (* Loop for stat summation *)
    FOR i := 0 TO N-LowerBound DO
        Xs := X[i]; Ys := Y[i];
        Sum := Sum + 1.0;
        SumX := SumX + Xs;
        SumY := SumY + Ys;
        SumXX := SumXX + Xs * Xs;
        SumYY := SumYY + Ys * Ys;
        SumXY := SumXY + Xs * Ys;
    END;
    (* Calculate intermediate results *)
    MeanX := SumX / Sum;
    MeanY := SumY / Sum;
    SdevX := sqrt((SumXX - SumX * SumX / Sum)/(Sum - 1.0));
    SdevY := sqrt((SumYY - SumY * SumY / Sum)/(Sum - 1.0));
END Regression;

PROCEDURE Slope() : REAL;
(* Function that returns the slope of the best fit line *)

BEGIN
    IF Sum > 1.0 THEN
        RETURN (SumXY - MeanX * MeanY * Sum) / (SdevX * SdevX * (Sum - 1.0))
    ELSE RETURN 0.0 (* default value for insufficient data *)
    END;
END Slope;

PROCEDURE Intercept() : REAL;
(* Function that returns the intercept of the best fit line *)
BEGIN
    IF Sum > 1.0 THEN
        RETURN MeanY - Slope() * MeanX
    ELSE RETURN 0.0 (* default value for insufficient data *)
    END;
END Intercept;

PROCEDURE R2() : REAL;
(* Function that returns the goodness of the best fit line *)

VAR R : REAL;

BEGIN
    IF Sum > 1.0 THEN
        R := SdevX / SdevY * Slope();
        RETURN R * R
```

(continued on page 99)



QNX: Other operating systems may never lift the PC to such heights.

QNX was the first operating system with multi-tasking, multi-user capabilities for the IBM PC and compatibles (1982). QNX was first to achieve the same capabilities for the IBM AT and compatibles. Then, in 1984, QNX achieved integrated networking capability, something no other multi-user, multi-tasking operating system has achieved, or is likely to achieve, on IBM personal computers.

QNX consistently beats competitive operating systems to each new height in PC performance. Unlike other multi-user, multi-tasking opeating systems, QNX is *not* UNIX-based. Being rigid and monolithic, UNIX architecture is difficult to adapt. By contrast, QNX is modular, based on message-passing, and easy to adapt. UNIX developers have had to preoccupy themselves with merely getting UNIX to work on the personal computer, while QNX,designed specifically for this environment, has enjoyed rapid and continuous growth.

QNX's superb performance and compact size is the result of one dedicated design team with a common purpose and complete understanding of both the software and the environment in which it must run. Quick and efficient on a PC, QNX soars on an AT.

Of all multi-user, multi-tasking operatings systems, only QNX provides unrestricted communications among all tasks anywhere on the network. Therefore, only QNX can implement true distributed processing, distributed file systems, and distributed devices, with no need for central file servers.

No one wants to abandon the tremendous amount of DOS software available. QNX does not force that choice. DOS 2.1 and 3.1 run as single-user tasks under QNX, even on the AT in protected mode.

With over 22,000 systems installed worldwide, QNX remains unsurpassed in PC performance.

End Users, VAR's, OEM's and Software Developers – ask us how QNX can add value to your products!

Multi-User	10 serial terminals per PC, AT.	**C Compiler**	Standard Kernighan and Ritchie.
Multi-Tasking	40 (64) tasks per PC (AT).	**Flexibility**	Single PC, networked PC's, single PC with terminals, networked PC's with terminals. No central servers. Full sharing of disks, devices and CPU's.
Networking	2.5 Megabit token ring. 255 PC's and/or AT's per network. 10,000 tasks per network. Thousands of users per network.		
Real Time	2800 task switches/sec (AT).	**PC-DOS**	PC-DOS runs as a QNX task.
Message Passing	Fast intertask communication between tasks on any machine.	**Cost**	From US $450. Runtime pricing available.

For further information or a free demonstration diskette, please telephone **(613) 726-1893.**

The only multi-user, multi-tasking, networking, real-time operating system for the IBM PC, AT and compatibles.

By Quantum Software.

Quantum Software Systems Ltd., Moodie Drive High Tech Park, 215 Stafford Road, Ottawa, Ontario, Canada K2H 9C1

Circle **no. 158** on reader service card.

Listing Four

(Listing continued, text begins on page 108.)

```
     ELSE RETURN 0.0 (* default value for insufficient data *)
     END;
END R2;

BEGIN
(* Initilaize inner module by setting stat summation equal to zero *)
     Sum := 0.0; SumXY := 0.0;
     SumX := 0.0; SumXX := 0.0;
     SumY := 0.0; SumYY := 0.0;
END InnerWorking;

END BestFit.
```

End Listing Four

Listing Five

```
Listing  5. Turbo Pascal program to demonstrate the first method for
external menu storage.

program test_method1;

(* Program to test first method for external menu storage *)

TYPE
      STRING14 = STRING[14];
      STRING80 = STRING[80];
      Screen_Image = ARRAY [0..24] OF STRING80;

VAR   Shift_Row, Shift_Col, Screen_Line_Count : INTEGER;
      Screen_Line : Screen_Image;
      MenuFile : STRING14;

PROCEDURE Read_Menu(Menu_Filename : STRING14;
                    VAR Shift_Row, Shift_Col,
                        Screen_Line_Count : INTEGER;
                    VAR Screen_Line : Screen_Image);
(* Procedure to read menu image from text file.  If file is *)
(* nonexistant the program will halt.                       *)

CONST MAX_SYMBOL = 255;

TYPE CharSet = Set OF CHAR;
     Symbol_Table = ARRAY [0..MAX_SYMBOL] OF INTEGER;


VAR FileVar : TEXT;
    Line : STRING80;
    Table : Symbol_Table;
    I, K, Error_Code   : INTEGER;
    Symbol_Char : CHAR;
    Operation_Set : CharSet;
    Duplicate : BOOLEAN;

(*---------------------------------------------------------*)

PROCEDURE INC(VAR A : INTEGER);
(* Increment integer by one *)
BEGIN
     A := A + 1
END; (* INC *)

(*---------------------------------------------------------*)


PROCEDURE Upcase_Str(VAR S : STRING80);
(* Convert string to upercase *)
VAR I : INTEGER;
BEGIN
     FOR I := 1 TO Length(S) DO
          S[I] := Upcase(S[I]);

END; (* Upcase_Str *)

(*---------------------------------------------------------*)


FUNCTION Extract_Number(Line : STRING80; Skip : INTEGER;
                        VAR ErrorCode : INTEGER) : INTEGER;
(* Function to extract an integer from a text line *)
VAR J : INTEGER;

BEGIN
     IF Skip > 0 THEN Delete(Line,1,Skip); (* Remove chars from string *)
     (* Remove blanks *)
     WHILE Line[1] = ' ' DO
          Delete(Line,1,1);
     (* END WHILE *)
     Line := Line[1] + Line[2] + Line[3];
     VAL(Line,J,Error_Code);
     Extract_Number := J
END; (* Extract_Number *)


(*---------------------------------------------------------*)

PROCEDURE Build_Screen(Line : STRING80;
                       VAR Screen_Line_Count : INTEGER;
                       VAR Screen_Line : Screen_Image);

VAR J : INTEGER;
    Ch : CHAR;

BEGIN
     IF Length(Line) > 0 THEN BEGIN
          FOR J := 1 TO Length(Line) DO BEGIN
               Ch := Line[J];
               IF Ch IN Operation_Set THEN
                  Line[J] := CHR(Table[ORD(Ch)]);
          END; (* FOR *)
          Screen_Line[Screen_Line_Count] := Line;
          INC(Screen_Line_Count);
     END;
END; (* Build_Screen *)
```

(continued on next page)

Listing Five **(Listing continued, text begins on page 108.)**

```
BEGIN
    Assign(FileVar, Menu_Filename);
    Reset(FileVar);
    IF (IOResult = 0)
    THEN BEGIN
        Operation_Set := ['!','@','#','$','%','^','&','/','\','|','-','_'];
        (* Initialize screen line strings *)
        FOR I := 0 TO 24 DO
            Screen_Line[I] := '';
        (* Initialize symbol table entries *)
        FOR I := 0 TO MAX_SYMBOL DO
            Table[I] := I;

        (* Read first line *)
        READLN(FileVar, Line);
        Upcase_Str(Line);
        WHILE (NOT Eof(FileVar)) AND (Line <> 'START') DO BEGIN
            IF Line[1] IN Operation_set
            THEN BEGIN
                Symbol_Char := Line[1];
                K := ORD(Symbol_Char);
                Table[K] := Extract_Number(Line,1,Error_code);
                IF (Error_Code > 0) OR
                   (NOT (Table[K] IN [0..255])) THEN
                        Table[K] := Ord('*');
            END;
            (* Read next line *)
            READLN(FileVar, Line);
        END; (* WHILE *)

        Screen_Line_Count := 0;
        Shift_Col := 0;
        Shift_Row := 0;
        (* Read next line that may contain row/column offset *)
        FOR I := 1 TO 2 DO BEGIN
            READLN(FileVar, Line);
            Upcase_Str(Line);
            IF Pos('SHIFTROW',Line) > 0 THEN BEGIN
                Shift_Row := Extract_Number(Line,8,Error_Code);
                IF Error_Code > 0 THEN Shift_Row := 0;
            END
            ELSE IF Pos('SHIFTCOL',Line) > 0 THEN BEGIN
                Shift_Col := Extract_Number(Line,8,Error_Code);
                IF Error_Code > 0 THEN Shift_Col := 0;
            END
            ELSE Build_Screen(Line,Screen_Line_Count,Screen_Line);
        END; (* FOR *)

        WHILE NOT EOF(FileVar) AND (Screen_Line_Count < 25) DO BEGIN
            READLN(FileVar, Line);
            Build_Screen(Line,Screen_Line_Count,Screen_Line);
        END; (* WHILE *)
        Close(FileVar);
    END
    ELSE Halt;

END; (* Read_Menu *)


(*-----------------------------------------------------------------*)

PROCEDURE DISP_STR(S : STRING80; Row, Col : INTEGER);
(* Procedure to write a string to the screen memory *)

TYPE SCREEN80 = ARRAY [1..25,1..80,1..2] OF CHAR;

VAR MONODISP : SCREEN80 Absolute $B000:0000;
    COLODISP : SCREEN80 Absolute $B800:0000;
    I, J, Mode : INTEGER;

BEGIN
    J := Length(S);
    Mode := MEM[$0040:$0049];
    IF Mode IN [2..3] THEN
        FOR I := 1 TO J DO
            COLODISP[Row,Col + I - 1,1] := S[I];
    IF Mode = 7 THEN
        FOR I := 1 TO J DO
            MONODISP[Row,Col + I -1,1] := S[I];
END;

(*-----------------------------------------------------------------*)


PROCEDURE Show_Menu(VAR Shift_Row, Shift_Col, Screen_Line_Count : INTEGER;
                    VAR Screen_Line : Screen_Image);

VAR I : INTEGER;

BEGIN
    FOR I := 0 TO Screen_Line_Count DO
        DISP_STR(Screen_Line[I],(I+Shift_Row+1),(1+Shift_Col));

END; (* Show_Menu *)

BEGIN
   ClrScr;
   WRITE('Enter filename '); READLN(MenuFile); WRITELN;
   Read_Menu(MenuFile, Shift_Row, Shift_Col,
             Screen_Line_Count,  Screen_Line);
   Show_Menu(Shift_Row, Shift_Col,Screen_Line_Count,  Screen_Line);
   REPEAT UNTIL KeyPressed;
END.
```

End Listing Five

Listing Six

```
Listing 6. Turbo Pascal program to demosntrate the second method for
external menu storage.

program test_method2;

(* Program to test the second method for external menu storage *)

TYPE
      STRING14 = STRING[14];
      LSTRING = STRING[255];
      Screen_Image = ARRAY [0..24] OF LSTRING;

VAR  Shift_Row, Shift_Col, Screen_Line_Count : INTEGER;
     Screen_Line : Screen_Image;
     MenuFile : STRING14;

PROCEDURE Read_Menu(Menu_Filename : STRING14;
                    VAR Shift_Row, Shift_Col,
                        Screen_Line_Count : INTEGER;
                    VAR Screen_Line : Screen_Image);
(* Procedure to read menu image from text file.  If file is *)
(* nonexistant the program will halt.                        *)

CONST MAX_SYMBOL = 255;

TYPE CharSet = Set OF CHAR;
     Symbol_Table = ARRAY [0..MAX_SYMBOL] OF INTEGER;


VAR FileVar : TEXT;
    Line : LSTRING;
    Table : Symbol_Table;
    I, K, Error_Code,
    Upper_Left_Corner, Upper_Right_Corner, Lower_Left_Corner,
    Lower_Right_Corner, Horizontal_Line, Vertical_Line,
    Cross_Bar, Left_Tee, Right_Tee,
    Up_Tee, Down_Tee,
    Left_Edge, Right_Edge,
    Vertical_Frames, Horizontal_Frames, Frame_Code,
    Number : INTEGER;
    Symbol_Char : CHAR;

(*----------------------------------------------------------*)

PROCEDURE INC(VAR A : INTEGER);
(* Increment integer by one *)
BEGIN
    A := A + 1
END; (* INC *)

(*----------------------------------------------------------*)


PROCEDURE Upcase_Str(VAR S : LSTRING);
(* Convert string to uppercase *)
VAR I : INTEGER;
BEGIN
    FOR I := 1 TO Length(S) DO
        S[I] := Upcase(S[I]);

END; (* Upcase_Str *)

(*----------------------------------------------------------*)


FUNCTION Extract_Number(Line : LSTRING; Skip : INTEGER) : INTEGER;
(* Function to extract an integer from a text line *)
VAR J, SUM : INTEGER;

BEGIN
    IF Skip > 0 THEN Delete(Line,1,Skip); (* Remove chars from string *)
    (* Remove blanks *)
    WHILE Line[1] = ' ' DO
        Delete(Line,1,1);
    (* END WHILE *)
    SUM := 0;
    J := 1;
    WHILE (J <= Length(Line)) AND (Line[J] IN ['0'..'9']) DO BEGIN
        SUM := 10 * SUM + ORD(Line[J]) - ORD('0');
        INC(J)
    END;

    Extract_Number := SUM
END; (* Extract_Number *)


(*----------------------------------------------------------*)

FUNCTION Get_Char_Code(S : LSTRING) : INTEGER;
(* Function to interpret frame symbol and return its ASCII code *)

VAR I, ASCII_Code : INTEGER;

BEGIN
    IF S = 'ULC' THEN  ASCII_Code := Upper_Left_Corner
    ELSE IF S = 'URC' THEN  ASCII_Code := Upper_Right_Corner
    ELSE IF S = 'LLC' THEN  ASCII_Code := Lower_Left_Corner
    ELSE IF S = 'LRC' THEN  ASCII_Code := Lower_Right_Corner
    ELSE IF S = 'HLN' THEN  ASCII_Code := Horizontal_Line
    ELSE IF S = 'VLN' THEN  ASCII_Code := Vertical_Line
    ELSE IF S = 'CRS' THEN  ASCII_Code := Cross_Bar
    ELSE IF S = 'LFT' THEN  ASCII_Code := Left_Tee
    ELSE IF S = 'RTT' THEN  ASCII_Code := Right_Tee
    ELSE IF S = 'UPT' THEN  ASCII_Code := Up_Tee
    ELSE IF S = 'DNT' THEN  ASCII_Code := Down_Tee
    ELSE ASCII_Code := ORD('-'); (* error value return 'A' *)
    Get_Char_Code := ASCII_Code;
END; (* Get_Char_Code *)


(*----------------------------------------------------------*)

PROCEDURE Build_Screen(Line : LSTRING;
                       VAR Screen_Line_Count : INTEGER;
                       VAR Screen_Line : Screen_Image);

VAR I, J, K, Long, Count : INTEGER;
    Ch, Symbol : CHAR;
    Build_Line, Sub_String : LSTRING;


BEGIN
    IF Length(Line) > 0 THEN BEGIN
        J := 1;
        Long := Length(Line);
        Build_Line := '';
        Count := 0;
        WHILE J <= Long DO BEGIN
            Ch := UpCase(Line[J]);
            CASE Ch OF
                '@' : BEGIN
```

(continued on page 103)

Brand New From Peter Norton
A PROGRAMMER'S EDITOR

that's *lightning fast* with the *hot* features programmers need

only **$50**

Direct from the man who gave you *The Norton Utilities, Inside the IBM PC,* and the *Peter Norton Programmer's Guide.*

THE NORTON EDITOR™

"This is the programmer's editor that I wished I'd had when I wrote my *Norton Utilities.* You can ***program your way to glory*** with *The Norton Editor.*"

Peter Norton

***Easily customized,* and saved**
Split-screen editing
A wonderful condensed/outline display
***Great for* assembler, Pascal *and* C**

Peter Norton Computing, Inc., 2210 Wilshire Boulevard, Santa Monica, CA 90403, 213-453-2361. Visa, Mastercard and phone orders welcome.

The Norton Editor™ is a trademark of Peter Norton Computing, Inc. © 1986 Peter Norton Computing.

Circle **no. 243** on reader service card.

The C Programmer's Assistant

C TOOLSET™

UNIX-like Utilities for Managing C Source Code

No C programmer should be without their assistant — C ToolSet. All of the utility programs are tailored to support the C language, but you can modify them to work with other languages too.

Source code in standard K&R C is included; and you are welcome to use it with any compiler (UNIX compatible) and operating system you choose.

12 Time Savers

DIFF - Compares text files on a line-by-line basis; use *CMP* for byte-by-byte. Indispensable for showing changes among versions of a program under development.
GREP - Regular exp. ssion search. Ideal for finding a procedural call or a variable definition amid a large number of header and source files.
FCHART - Traces the flow of control between the large modules of a program.
PP (C Beautifier) - Formats C program files so they are easier to read.
CUTIL - A general purpose file filter.

Requires MSDOS and 12K RAM

CCREF - Cross references variables used within a program.
CBC (curly brace checker) - checks for pairing of curly braces, parens, quotes, and comments.
Other utilities include *DOCMAKE, ASCII, NOCOM,* and *PRNT.*
Source code to every program is included!

Thorough User Support Text and Online

C ToolSet documentation contains descriptions of each program, a listing of program options (if any), and a sample run of the program.

On-line help gives you information on the programs and how to run them. Most of the programs respond to -? on the command line with a list of options.

Call 800-821-2492 to order C ToolSet risk-free for only $95.

Solution Systems™
335-D Washington St.,
Norwell, MA 02061
(617) 659-1571

Full refund if not satisfied in 30 days.

Circle **no. 152** on reader service card.

New Release!

SEIDL MAKE UTILITY
Version 2.0

The BEST just got BETTER!

If you're serious about software development, you need the **SEIDL MAKE UTILITY** (*SMK*). *SMK* increases productivity by cutting development time. When changes are made to a software system, *SMK* executes the minimum set of commands needed to rebuild the system.

- ***Powerful*** Ready to handle your biggest projects, but simple enough to use on your smallest.
- ***Fast*** Analyzes hundreds of dependencies in seconds.
- ***Flexible*** Total control over all generated commands.
- ***Expandable*** *SMK* integrates with the **SEIDL VERSION MANAGER** (*SVM*), a powerful source control system.
- ***Easy-to-Use*** Excellent user's manual, dependency definition language (DDL) and error diagnostics.
- ***Affordable*** MS-DOS(PC-DOS) version only ***$99.95*** plus ***$3.50*** p&h. Call about other operating systems.

"SMK is a very good Make indeed. Its major distinction is a truly simple Dependency Definition Language, easy to learn and easy to use... you'll probably bless SMK."
— Sextant Magazine, July 1986

"SMK offers many unique features. [The error handling facility] is extremely useful if a large number of files must be recompiled."
— Computer Language Magazine, June 1986

Call Today
1-313-662-8086

Visa/MC/COD Accepted
Dealer Inquiries Invited

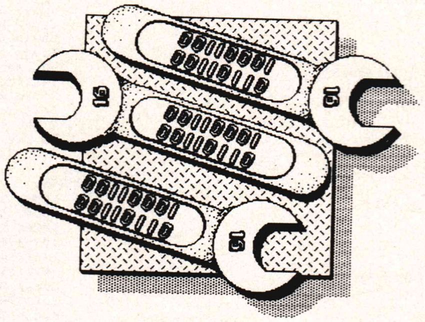
SEIDL COMPUTER ENGINEERING
3106 Hilltop Dr., Ann Arbor, MI 48103

Circle **no. 114** on reader service card.

Listing Six (Listing continued, text begins on page 108.)

```
                Sub_String := '';
                FOR I := 1 TO 3 DO
                    Sub_String := Sub_String + Line[J+I];
                J := J + 3; (* advance character pointer *)
                Symbol := CHR(Get_Char_Code(Sub_String));
                Build_Line := Build_Line + Symbol;
                INC(Count);
              END;
        'D' : BEGIN (* Duplicate a frame character *)
                Sub_String := Line[J+1] + Line[J+2] + Line[J+3];
                J := J + 4; (* advance character pointer *)
                Symbol := CHR(Get_Char_Code(Sub_String));
                Sub_String := Line[J] + Line[J+1];
                J := J + 1;
                K := Extract_Number(Sub_String,0);
                IF (K > 0) THEN BEGIN
                    Count := Count + K;
                    FOR I := 1 TO K DO
                        Build_Line := Build_Line + Symbol;
                END; (* IF *)
              END;
        'S' : BEGIN (* Skip # column positions *)
                Sub_String := Line[J+1] + Line[J+2];
                J := J + 2; (* advance character pointer *)
                K := Extract_Number(Sub_String,0);
                IF (K > 0) THEN BEGIN
                    Count := Count + K;
                    FOR I  := 1 TO K DO
                        Build_Line := Build_Line + ' ';
                END; (* IF*)
              END;
        '"' : BEGIN (* Display text *)
                INC(J);
                WHILE (Line[J] <> '|') AND (J <= Long) DO BEGIN
                    Build_Line := Build_Line + Line[J];
                    INC(J); INC(Count)
                END; (* WHILE *)
                Count := COunt - 1;
              END;
        '#' : BEGIN
                Sub_String := Line[J+1] + Line[J+2];
                J := J + 2; (* advance character pointer *)
                K := Extract_Number(Sub_String,0);
                IF (K < Right_Edge) AND (Count < K) THEN BEGIN
                    FOR I := 1 TO K - Count DO
                        Build_Line := Build_Line + ' ';
                    Count := K;
                      END; (* IF *)
                    END;
                'V' : BEGIN (* Draw vertical edges *)
                        Build_Line := CHR(Vertical_Line);
                        FOR I := Left_Edge+1 TO Right_Edge-1 DO
                            Build_Line := Build_Line + ' ';
                        Build_Line := Build_Line + CHR(Vertical_Line);
                      END;
                'H' : BEGIN (* Draw horizontal edge *)
                        Symbol := CHR(Horizontal_Line);
                        FOR I := Left_Edge+1 TO Right_Edge-1 DO
                            Build_Line := Build_Line + Symbol;
                      END;
              END; (* CASE *)
              INC(J);
              WHILE Line[J] = ' ' DO   INC(J);
          END; (* FOR *)
          Screen_Line[Screen_Line_Count] := Build_Line;
          INC(Screen_Line_Count);
      END;
  END; (* Build Screen *)
  BEGIN
      Assign(FileVar, Menu_Filename);
      (*$I-*) Reset(FileVar); (*$I+*)
      IF (IOResult = 0)
      THEN BEGIN
          (* Initialize screen line strings *)
          FOR I := 0 TO 24 DO
              Screen_Line[I] := '';
          Left_Edge := 1;
          Right_Edge := 80;
          Vertical_Frames := 2;
          Horizontal_Frames := 2;

          (* Read first line *)
          READLN(FileVar, Line);
          Upcase_Str(Line);
          WHILE (NOT Eof(FileVar)) AND (Line <> 'START') DO BEGIN
              Symbol_Char := Line[1];
              K := ORD(Symbol_Char);
              IF Symbol_Char IN ['R','L','H','V'] THEN BEGIN
                  Number := Extract_Number(Line,1);
                  IF (Error_Code = 0) THEN
                      CASE Symbol_Char OF
                          'R' : Right_Edge  := Number;
                          'L' : Left_Edge  := Number;
                          'H' : IF (Number IN [1..2]) THEN
                                    Horizontal_Frames := Number;
                          'V' : IF (Number IN [1..2]) THEN
                                    Vertical_Frames := Number;
                      END; (* CASE *)
              END; (* IF *)
              (* Read next line *)
              READLN(FileVar, Line);
          END; (* WHILE *)

          (* Check edges *)
          IF (Right_Edge - Left_Edge) <= 4 THEN BEGIN
              Left_Edge := 1;
              Right_Edge := 80;
          END; (* IF *)

          Frame_Code := 10 * Horizontal_Frames + Vertical_Frames;
          (* Select frame type *)
          CASE Frame_Code OF
              11 : BEGIN
                      Upper_Left_Corner := 218;
                      Upper_Right_Corner := 191;
                      Lower_Left_Corner := 192;
                      Lower_Right_Corner := 217;
                      Horizontal_Line := 196;
                      Vertical_Line := 179;
                      Cross_Bar := 197;
                      Left_Tee := 195;
                      Right_Tee := 180;
                      Up_Tee := 193;
                      Down_Tee := 194;
                  END;
              12 : BEGIN
                      Upper_Left_Corner := 214;
                      Upper_Right_Corner := 183;
                      Lower_Left_Corner := 211;
                      Lower_Right_Corner := 189;
                      Horizontal_Line := 196;
                      Vertical_Line := 186;
                      Cross_Bar := 215;
                      Left_Tee := 199;
                      Right_Tee := 182;
                      Up_Tee := 208;
                      Down_Tee := 210;
                  END;
              21 : BEGIN
                      Upper_Left_Corner := 213;
                      Upper_Right_Corner := 184;
                      Lower_Left_Corner := 212;
                      Lower_Right_Corner := 190;
                      Horizontal_Line := 205;
                      Vertical_Line := 179;
                      Cross_Bar := 216;
                      Left_Tee := 198;
                      Right_Tee := 181;
                      Up_Tee := 207;
                      Down_Tee := 209;
                  END;
              22 : BEGIN
                      Upper_Left_Corner := 201;

                          Upper_Right_Corner := 187;
                          Lower_Left_Corner := 200;
                          Lower_Right_Corner := 188;
                          Horizontal_Line := 205;
                          Vertical_Line := 186;
                          Cross_Bar := 206;
                          Left_Tee := 204;
                          Right_Tee := 185;
                          Up_Tee := 202;
                          Down_Tee := 203;
                      END;
          END; (* CASE *)

          Screen_Line_Count := 0;
          Shift_Col := 0;
          Shift_Row := 0;
          (* Read next line that may contain row/column offset *)
          FOR I := 1 TO 2 DO BEGIN
              READLN(FileVar, Line);
              Upcase_Str(Line);
              IF Pos('SHIFTROW',Line) > 0 THEN BEGIN
                  Shift_Row := Extract_Number(Line,8);
                  IF Error_Code > 0 THEN Shift_Row := 0;
              END
              ELSE IF Pos('SHIFTCOL',Line) > 0 THEN BEGIN
                  Shift_Col := Extract_Number(Line,8);
                  IF Error_Code > 0 THEN Shift_Col := 0;
              END
              ELSE Build_Screen(Line,Screen_Line_Count,Screen_Line);
          END; (* FOR *)

          WHILE NOT EOF(FileVar) AND (Screen_Line_Count < 25) DO BEGIN
              READLN(FileVar, Line);
              Build_Screen(Line,Screen_Line_Count,Screen_Line);
          END; (* WHILE *)
          Screen_Line_Count := Screen_Line_Count - 1;
          Close(FileVar);
      END
      ELSE BEGIN
          WRITE(^G^G);
          Halt;
      END;
END; (* Read_Menu *)
(*---------------------------------------------------------------*)

PROCEDURE DISP_STR(S : LSTRING; Row, Col : INTEGER);
(* Procedure to write a string to the screen memory *)

TYPE SCREEN80 = ARRAY [1..25,1..80,1..2] OF CHAR;

VAR MONODISP : SCREEN80 Absolute $B000:0000;
    COLODISP : SCREEN80 Absolute $B800:0000;
    I, J, Mode : INTEGER;

BEGIN
    J := Length(S);
    Mode := MEM[$0040:$0049];
    IF Mode IN [2..3] THEN
        FOR I := 1 TO J DO
            COLODISP[Row,Col + I - 1,1] := S[I];
    IF Mode = 7 THEN
        FOR I := 1 TO J DO
            MONODISP[Row,Col + I -1,1] := S[I];
END; (* DISP_STR *)

(*---------------------------------------------------------------*)

PROCEDURE Show_Menu(VAR Shift_Row, Shift_Col, Screen_Line_Count : INTEGER;
                    VAR Screen_Line : Screen_Image);

VAR I : INTEGER;
BEGIN
    FOR I := 0 TO Screen_Line_Count DO
        DISP_STR(Screen_Line[I],(I+Shift_Row+1),(1+Shift_Col));

END; (* Show_Menu *)
(*---------------------------------------------------------------*)

BEGIN (*-------------- M A I N ----------------*)
   ClrScr;
   WRITE('Enter filename '); READLN(MenuFile); WRITELN;
   Read_Menu(MenuFile, Shift_Row, Shift_Col,
          Screen_Line_Count,  Screen_Line);
   Show_Menu(Shift_Row, Shift_Col,Screen_Line_Count,  Screen_Line);
   REPEAT UNTIL KeyPressed;
END.
```

End Listings



Critical Error Handling

In the July 1986 16-Bit Toolbox column, I quoted a letter stating that Computer Innovations' C compiler, Version 1.31, has been placed in the public domain. This was and is completely incorrect, and although a retraction has already been printed elsewhere in *DDJ*, I wish to apologize to Computer Innovations for publishing this letter without checking the facts—major league stupidity on my part.

Comparing Strings

As the October column included two listings of 8086 string searching routines, I thought I'd continue the trend this month with 8086- and 68000-based string comparison routines (Listings One and Two, page 86). These procedures work very much like the *strcmp* library function in C does and return flags indicating whether the first string is less than, equal to, or greater than the second string. The routines are not completely symmetrical but do indicate how the 8086's special string instructions can provide a considerable space and speed advantage over the 68000's in some situations.

Changing the Master Environment

The October 1986 16-Bit Toolbox column included a brief introduction to environment blocks under MS-DOS, and at that time I promised to discuss several different ways to change the master environment block from an executing program. After considering the matter further, I have decided to break that promise, and this month I will present only one method (which I feel to be the safest and most portable).

by Ray Duncan

MS-DOS has an undocumented entry point, software interrupt *2eh*, which is called with registers *ds:si* pointing to a command string in the form of a count byte, followed by ASCII text, followed by a carriage return (which is not included in the count). This entry point appears to be a sort of back door to the command interpreter buried in COMMAND.COM, and if you pass it a string of the form *set name=parameter*, you will find that the system's master environment block will be modified rather than just your program's local environment block.

Although *int 2eh* is not discussed in any Microsoft reference or documentation I have seen (even the OEM adaptation guide or the new *MS-DOS Technical Reference Encyclopedia*), it seems to be present and work the same way in all PC-DOS/MS-DOS versions 2.0 through 3.2. Because Microsoft is now clearly focusing its operating system efforts on 80286-based protected mode versions of DOS, I suspect that the 8086/88-based versions 2 and 3 of DOS are going to become fairly static (except perhaps for bug fixes). Therefore, it should be safe to build calls to *int 2eh* into your application software for these DOS versions.

I have two listings this month to illustrate the use of *int 2eh*. The first (Listing Three, page 86) is from David Gwillim in Los Angeles and is in the form of a small macro assembler COM program. David has contributed many other interesting and useful utilities to bulletin boards in the Los Angeles area. The second example (Listing Four, page 88) is from Dan Lewis of Key Software Products, in Menlo Park, California, who contributed a (DeSmet) C routine called *Set_Var* that demonstrates use of the undocumented *int 2eh* function to change the master environment block. Dan is an associate professor of computer science at Santa Clara University.

File Handles

Ross Nelson of San Jose writes: "I have been following the file handles/redirection letters in your column, and I have an observation that you may find interesting. Although it seems like the *DUP* and *FDUP* functions (*int 21h*, functions *45h* and *46h*, respectively) are tailor-made for redirection (and I wouldn't be surprised if that is the approach Unix uses), that is not the method MS-DOS uses internally. I have included the C code for a routine called STD.C that emulates the method COMMAND.COM uses to do redirection (at least, this method was used in MS-DOS, Version 3.0, last time I checked).

"MS-DOS apparently keeps a set of 'actual' handles deep in the bowels of the operating system, with reference counts and so on. Each 'task' gets its own set of 'virtual' handles, which always begin with handle 0. The virtual-to-actual translation table is held in the program segment prefix for each task. The code I am supplying was written before I saw your May column, which pointed out that the table doesn't have to reside there because there is a pointer and a count in the PSP that points to the actual table. Anyway, when you issue a redirection command to COMMAND.COM, it doesn't bother with *DUP* and *FDUP*, it just mucks with the virtual-to-actual table in the PSP of the task that will run with redirection."

Ross' program accompanies this month's column as Listing Five, page 91.

File-name Wildcards

Randy Langer, of MicroSphere Technology in Chico, California, writes: "I am sending you an assembly-language C function [Listing Six, page 92] that performs wildcard file-name expansion under MS-DOS 2.x. Adapted

from the documentation presented by Peter Norton [*The Programmer's Guide to the IBM PC* (Redmond, Wash.: Microsoft Press, 1985)], this function supplies a solution to a common problem in writing operating system utilities.

"To use this function, the user first defines a structure as *typedef*ed in wildcard.h [Listing Seven, page 94]. Then, for each *filespec* to be expanded, the user copies a pointer to the *filespec* into the structure and initializes the *flag* member to 0. Thereupon, the user repeatedly calls *wildcard()*, passing a pointer to the structure defined for this purpose. The function call will either return a pointer to the structure's *filename* member or *NULL* if no more matching file names exist. The example shown in wildtest.c [Listing Eight, page 94] will clarify this.

"Assuming the function returns the file-name pointer, other data regarding the file are also available from the call. The structure member *f_atts* contains the 'attributes' of the file (read-only, subdirectory, and so on; see Norton, page 116). The member *datetime* contains the file's time/date stamp in a format that is compatable with the Aztec C *time* library functions. The *filesize* member contains what its name suggests. The final member, *tempdata*, contains link data for subsequent *search next* calls, so keep yer mitts offa it.

"It should be noted that the returned file name does not include the drive/path involved; this must be divined from the file spec. The algorithm in wildtest.c shows a method for doing this."

New Books to Buy

Strauss, Edmund. *Inside the 80286.* New York: Brady Communications Co., 1986.

Edmund Strauss is a senior application marketing engineer for the 80286 microprocessor and clearly knows his stuff. This book is a very lucid presentation of the special features of the 80286, including its additional registers and instructions, protected virtual-addressing modes, task switching, and privilege levels. The book includes many excellent diagrams and several lengthy and well-commented assembly-language programming examples, and it even finishes up with

16-BIT
(continued from page 105)

schematics and instructions to build your own small 80286-based system—highly recommended.

Concurrent DOS

Henry Velick of Monsey, New York, writes: "After reading your July column, and Mr. Mullan's contributions in particular, here are a few thoughts to add to the raging Concurrent PC-DOS controversy. The first is a question: how many useful (usable, well-used, common, and so on) PC programs will support a dumb ASCII terminal? Answer: not too many. As far as I know, most of the big-selling business programs, such as Lotus 1-2-3, dBASE III, Microsoft Word, and Crosstalk, are inextricably tied to the PC keyboard and display adapters. How useful, then, can a multiuser system be that supports only one keyboard/display, no matter how many dumb terminals it might support?

"There is, by the way, a reasonable alternative. A small company in New York state, called ANEX Technology, has some hardware/software products that allow up to four users on a PC or PC/XT and up to eight on a PC/AT. Unlike with Concurrent PC-DOS, each user has full use of a genuine PC keyboard (or aftermarket equivalent) and a real PC display (MDA, CGA, Hercules, Tecmar, and so on). Very few programs cannot run on this system. Those that can't require certain hard-wired memory locations to be available—most of these are games. But I have seen an XT with four monitors simultaneously running 1-2-3, a BASICA graphics demo, WordStar, and a screen-oriented data-entry program with very little degradation of performance—very impressive.

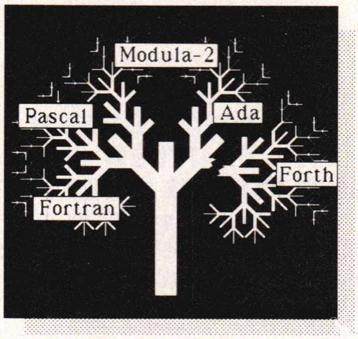
"ANEX's products are called MPC-4 and MPC-8 for the PC and AT, respectively. There is also a new low-end version called PC-ANEX that supports only two users. And although certainly more expensive than a copy of Concurrent PC-DOS, they are also a good deal cheaper than the additional PCs. No, I don't work for ANEX."

DDJ

(Listings begin on page 86.)

Vote for your favorite feature/article.
Circle Reader Service **No. 6.**

Turbo Pascal Procedural Parameters, Local Modules in Modula-2



In this issue, I will discuss implementing procedural parameters in Turbo Pascal, local modules in Modula-2, and two methods for modifying menus of an application program without editing it.

Procedural Parameters

Turbo Pascal does not support procedural parameters. Procedural parameters enable programmers to write applications that can easily employ alternate methods. For example, a program sorting several records can examine the size of the data and determine which sorting method to use. If there are few records, a bubble sort is applied; for a moderate number of records, the Shell-Metzner sorting method is used; and for a large data set, quicksort is employed.

Mike Babulic, of Calgary, Canada, has implemented procedural parameters; he offers three versions that differ in speed and reliance on Pascal and machine language. You can download them from the Borland SIG on CompuServe, where they are stored as an archive file. You will need the DEARC.PAS utility, available in the same SIG, to unpack all Mike's files.

The archive file contains six files:

1. ProcParm.PAS—Mike's documentation with his example program.
2. ProcParm.INC—Contains the *call_ProcParm* procedure.
3. ProcParm.QK—In-line code to be used instead of the *call_ProcParm* procedure. Quicker execution, but larger programs.

by Namir Clement Shammas

4. ProcParm.BIN—The third (and best) alternative, an externally assembled binary file. It is noted for fastest execution speed because of less stack manipulation and the least code added to the program (12 bytes!).
5. ProcParm.P—This is the "glue" that you include in a program so it can use ProcParm.BIN.
6. ProcParm.ASM—The source code for ProcParm.BIN.

The source code accompanying this column (Listings One and Two, page 96) includes that for the first two methods only.

The three versions to implement procedural parameters are:

1. The *call_ProcParm* procedure—This procedure is written in Turbo Pascal with no in-line code. When *call_ProcParm* is called by a procedure or function, it causes a short jump to the procedure whose offset is the last parameter of the calling procedure. This is done by swapping the last parameter, a pointer to a procedure, and the return address of the calling routine on the stack.

Using *call_ProcParm*, procedures are passed as parameters to other procedures. This is accomplished by defining a dummy procedure or function with exactly the same arguments as the procedure to be passed, except that a final added *integer*-type parameter is also defined. When the calling procedure invokes the dummy procedure, it passes the offset of the procedure to be executed in this last parameter. The *Ofs* function is frequently used for this purpose.

ProcParm returns to the routine specified by the last parameter of the dummy routine, and the stack will look exactly the same as if the "returned to" procedure had been called by the procedure that called the dummy one.

2. The in-line code method (ProcParm.QK)—You can *include* the ProcParm.QK file in your dummy procedure instead of calling the *ProcParm* procedure. This speeds up a program because fewer 8088 instructions are executed for jumping to the passed procedure. Be aware, however, that the cost of this speed is a larger program.

The *ProcPtr* type need not be declared if you use ProcParm.QK—the *integer* type is employed instead.

3. ProcParm.BIN and ProcParm.P—The ProcParm.P file is *include*d in your program to allow it to use the contents of the ProcParm.BIN file. As in the previous two methods, a dummy procedure is created with the procedure to be run as the last parameter. This is where the similarity ends.

ProcParm.P allows you to call *FAR* procedures and functions that lie outside the body of the program. In addition, it allows your programs to call procedures and functions in the body of your Turbo program (*NEAR* calls).

You can also make a *FAR* call with an offset. This is useful if you have set aside an area of storage for a group of procedures or functions. The method used is quite a bit slower than the standard *FAR* call. If speed is important, you should save the address of each procedure in a variable and use the regular *FAR* call. Mike has included a function to make life easier for you.

One problem with the *FAR* calls that Mike hasn't been able to solve yet is that you can't make them from inside the Turbo environment. You have to create a .COM file and execute that. If anyone comes up with a workable solution, please let Mike know.

Listing Three, page 96, shows an example I wrote using the first two methods. The program creates an ordered array of integers and reverses their order. In each method I have used the Shell sort and quicksort tech-

niques. The sample program also incorporates a routine to report the time at the beginning and end of a sorting method. This should give you a feel for the speed of each of Mike's routines.

Local Modules in Modula-2

The Modula-2 language supports local modules, a valuable feature that has received little attention. Local modules are modules that are nested inside others. Their interface with the parent module is established using import and export lists. The interface of local modules is very strict—for example, if the local module needs to write a string to the screen, it must explicitly import the *WriteString* procedure from the parent module. This clearly differs from the interface of ordinary Modula-2 procedures. The export lists contains all the items exported by the local module.

The advantages of local modules are twofold. First, the variables used in local modules are static; their values are retained between calls to the local modules. The second advantage is that local modules can initialize their static variables. This is done once, the first time the local module is called.

The advantage of using local modules, especially within library modules, is to hide some details concerning intermediate variables involved in certain calculations. One example, often mentioned in Modula-2 textbooks, is the generation of random numbers. The local module responsible for returning a random number is assigned the task of initializing the seed value. Its value is preserved by a local static variable between calls to the local module.

Listing Four, page 97, shows an example of using a local module inside a library module. The library module *BestFit* provides routines for a linear regression between two arrays, *X* and *Y*. The module exports functions that return the slope, intercept, and coefficient of determination statistics. The local module *InnerWorking* initializes the statistical summations, contains the *Regression* procedure to update them with data, and calculates intermediate results. The variables within the local module are both static and invisible to an application program employing module *BestFit*. This gives more of a black-box effect, in which intermediate data is retained by the library module and the final results are passed back to the calling application.

Local modules and their static variables can be used in a variety of other applications. A library module, for example, manipulates a stack of items and protects it from corruption. Procedures and functions are used to push, pop, rotate, and swap data items in the stack. The application program performs data transfer with static variables in the library module.

Modifying Menus Without Program Editing

Menus are employed in "friendly" application programs. Aesthetically appealing menus are placed in single- or double-line frames using extended ASCII characters. Normally, these programs contain the code for the menus. This means that if you want to alter the menu text, you must edit and recompile the source code. An alternative method is to store the menu text in a separate file. At run

TURBO
SCREEN/APPLICATION GENERATOR
BE 3 to 6 TIMES MORE PRODUCTIVE

For ~~$99.95~~ receive **6 Floppy Disks & Manual. DDJ Special: $69.95**

Expires 12/31/86

TURBO ISAM MASTER

TURBO MASTER helps you develop your functional specs and then allows you to QUICKLY prototype a validation model of your system. TURBO MASTER can then generate a super-fast Turbo Pascal Program that features advanced screen input and control, a professional control menu, and database functions.

- **MASTER DATABASE PROGRAM** - Generates Pascal Program Code for the following functions
  - —**Add a Record** - Allows both duplicate & unique only keys. Change the up to 48 fields comprising the primary key and the up to 7 secondary keys at any time during input, and the validity of the keys are checked and the ISAM key files are automatically adjusted.
  - —**Delete a Record** - Shows the record to be deleted on the screen and allows you to change your mind about deleting, and then adjusts the keys files automatically for the up to 8 keys. The disk space is reused automatically by the programs.
  - —**Edit a Record** - with a key change allowed.
  - —**Search Database by Key** - The ability to display the keys on the screen or printer in sorted order.
- **DATA BASE RECOVERY PROGRAM.** This program recovers your database if it corrupted by a power outage or certain hardware failures.
- **MULTIPLE ISAM FILES** - Can be used at the same time in a single program. The generator also produces context sensitive instructions on how to integrate the generated program into your main program.
- **DOCUMENTATION** - Print screens and ISAM specifications. Also inline program documentation is generated.

TURBO MENU MASTER

GENERATES 'Ready to Run' Turbo Programs with Dedicated Screens or Windows.

- **FEATURES:** Has Menu Data Base so it's easy to modify menus.

  PROVIDES SELECTION BY:
  - —Press of a number
  - —Press of a function key
  - —Press the high lighted letter
  - —Use the arrow keys

  ADAPTIVE SCREEN COLORS — Different screen colors are automatically used for color or monochrome display. This allows you to provide a better interface for the user.
- **INTERACTIVE MENU BUILDER**
  - —Allows for the automatic entry and reorder of selections
  - —Offers easy color selection
  - —Allows for choice of procedure, chain, execute or comment code generation for each selection
  - —Has startup Menu that YOU design.
  - —Provides for the integrated Display and control of the Key Lock Status.

TURBO RESIDENT SCREEN CAPTURE

which allows you to capture Text Screens from any running program.

TURBO SCREEN MASTER

GENERATES 'Ready to Run' Mulit-screen programs with advanced field definition.

- **USER SPECIFIED PROCEDURES** can be called before and after data entry for each field. This allows **SCREEN GENERATION THAT IS INDEPENDENT OF USER INSERTED PROGRAM STATEMENTS.**
- **YOU CAN CUSTOM PROGRAM** special field edits and processes which Screen Master automatically includes into the screen program. This allows for "On the Fly **FUNCTION KEY AND WINDOW ROUTINES.**
- **RECORD FORMATS** and initialization routines for Structures and Arrays are automatically generated.
- **CAN SPECIFY** a Protected Field to be automatically redisplayed if its value changes.
- **TYPES INCLUDE** Strings, Yes/No, Time, Male/Female and Numbers.
- **FIELDS CAN BE STORED** (declared) as Boolean, Integer, Real, Character, or String.

COMPARISON WITH OTHER PRODUCTS	TURBO SCREEN MASTER	SCREEN SCULPTURE Ver - 1.01
Full support for structures, arrays and Declarations specifications	YES	NO
Full support for user written procedures, function keys & help screens	YES	NO
Etch-A-Sketch Border Drawings	YES	NO
Point and Paint color interaction	YES	YES
Border color control	YES	NO
User defined valid character sets	YES	NO
Display of Caps/Num Lock Status	YES	NO
Optional realtime initialization of data/time	YES	NO

TURBO RESIDENT ISAM

Replaces Borland's Toolbox ISAM method

- Eliminates 8k of code space that Borland's Turbo Toolbox ISAM requires in your program.
- Puts the Key Buffer Data Area outside of your program's data space.
- Speeds Compling
- Eliminates errors caused by inconsistant ISAM specifications between ISAM files.
- Increases program loading time.
- Reduces object program size.
- Offers SUPER-FAST record creation and lookup. For example, it can CREATE over 20 single key records PER SECOND on an IBM AT running at a 9 meg clock rate.

OUR USERS REPORT

- *"Since Fall of 85, I have generated over 300 program modules with it and find it to be just what I needed. Most of all the modules represent 5000 to 8000 lines of Pascal Code"* Oner Systems
- *"By being able to produce a 21 screen and menu control demo so quickly helped me obtain the contract."*
- *"Speeded up my screen development by 6 times"* Elexor Associates.
- *"Has many of the features of the Super Mini development tools costing $10,000."* Applied Micro Systems.
- *"We developed 3 Vertical Market Applications in the 6 months we had your system."* Absolute Systems.

★★★★★★★★★★★★★★★★★

BTRIEVE INTERFACE MODULE

Allows full multiuser record locking and Automatic file recovery for the industry's most popular LANs. Works with the industry's leader of professional databases for multiuser LAN's (optional). **$99.95**

Requires Btrieve by SoftCraft, Inc.

★★★★★★★★★★★★★★★★★

TURBO MASTER TOOLS

The Turbo Toolkit Master includes procedures for full control over all screen attributes, advanced string functions, automatic control of multiple help screens, saving and retrieving screens to RAM buffers, Caps/Num/Scroll/Lock control procedures, and report procedures.

RISK FREE TRIAL

Try the demo package included for 30 days. If not pleased return for a full refund.

Credit Card Orders Call:

1-800-821-9503

In Florida (800) 342-0137

For Other Information Call (305) 892-5686

NO ROYALTIES on Generated Programs

YES, ENCLOSED IS **$69.95**

For

Turbo Master Tools
Turbo Menu Master
Turbo Screen Master
Turbo ISAM Master
Turbo Screen Capture
Resident ISAM

Systems Requires: IBM PC,XT,AT or 100% Compatible -246K MS DOS 2.0 or Higher, Turbo Pascal 3.0 - 2 DD/DS Disk Drives.

US orders add $7.50 S&H

All foreign orders add $15 per product ordered.

Name: ________________ **Phone:** ________________

Shipping Address: ________________

City: ________________

State: ________________

Zip ________________

VISA or MC #: ________________

Exp. Date ________________

C.O.D.'s will be accepted. Outside USA: make payment by bank draft, payable in US dollars drawn on a US bank.

HAWAIIAN VILLAGE COMPUTER SOFTWARE
1109 Pennsylvania Ave., St. Cloud, Florida 32769

ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. A COPY OF THIS WARRANTY IS AVAILABLE FREE, ON REQUEST.

DIGITAL RESEARCH COMPUTERS (214) 225-2309

TEXT TO SPEECH BOARD!

PC/XT COMPATIBLE. MAKE YOUR COMPUTER TALK!

A VERY POWERFUL AND AMAZING SPEECH CARD. USES THE NEW GENERAL INSTRUMENTS SPO256-AL2 SPEECH CHIP AND THE CTS256A-AL2 TEXT TO SPEECH CONVERTER.

THIS BOARD USES ONE SLOT ON THE MOTHER-BOARD AND REQUIRES A COM SERIAL PORT. BOARD MAY ALSO BE USED IN A STAND ALONE ENVIRONMENT WITH ALMOST ANY COMPUTER THAT HAS A RS232 SERIAL PORT. FEATURES ON BOARD AUDIO AMP OR MAY BE USED WITH EXTERNAL AMPS.

NEW!

DEMONSTRATION SOFTWARE AND A LIBRARY BUILDING PROGRAM ARE INCLUDED ON A 5¼ INCH PC/XT DISKETTE. FULL DOCUMENTATION AND SCHEMATICS ARE ALSO INCLUDED.

69.95 COMPLETE KIT — **89.95** ASSEMBLED & TESTED

HARD DISK CONTROLLER CARD

THE FANTASTIC KONAN KDC-230. WITH MANUAL. SUPER EASY TO CONFIGURE AND USE.

HALF SIZE PCB. $99.00 — ADD $8.00 FOR CABLES

MEX-PC MODEM SOFTWARE PACKAGE $49.95

A FANTASTIC COMMUNICATIONS PACKAGE WITH FEATURES TOO NUMEROUS TO LIST. SEE MAY/JUNE '86 MICRO SYSTEMS JOURNAL FOR FULL REVIEW OR CALL FOR BROCHURE. SUPPORTS COLOR, KERMIT PROTOCOL, Y MODEM BATCH, VT100 EMULATION, REMOTE OPERATION, ON LINE HELP, ETC. RUNS UNDER PC/MS DOS. 180 PAGE MANUAL & BINDER.

MEX-PACK LIST IS $99.95 — **SPECIAL $49.95**

HOT COMPUTER BOOKS!

GET THE MOST OUT OF THE INVESTMENT YOU MADE IN YOUR COMPUTER HARDWARE AND SOFTWARE WITH THE NEW WORDWARE SERIES OF COMPUTER REFERENCE AND TRAINING MANUALS. LEARN HOW TO MAKE ALL THAT EXPENSIVE SOFTWARE YOU BOUGHT REALLY COOK! *BUY ANY 3: TAKE AN EXTRA 10% DISCOUNT!*

ILLUSTRATED TURBO PASCAL	$13.50
ILLUSTRATED AUTOCAD	$18.50
ILLUSTRATED WORDSTAR	$18.50
ILLUSTRATED d BASE II	$13.95
ILLUSTRATED d BASE III	$18.50
ILLUSTRATED LOTUS 1-2-3	$15.50
ILLUSTRATED SYMPHONY	$17.50
ILLUSTRATED C PROGRAMMING	$17.95
ILLUSTRATED MS/PC DOS	$12.95
ILLUSTRATED MULTIMATE	$17.50
WORDSTAR CUSTOMIZING	$18.50
COMMUNICATIONS HANDBOOK	$12.50
ADVANCED TURBO PASCAL PROGRAMMING TECHNIQUES	$19.95

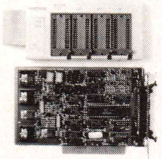
NEW!

S100 BUSS LIVES! WE STILL SELL S100 PRODUCTS AND CP/M SINGLE BOARDS. FREE CATALOG.

SCHEMA: Schematic Capture and Drawing Package

WORKS WITH PC'S AND COMPATIBLES. EASY TO USE WITH OFF THE SHELF PRINTERS, MOUSES, VIDEO CARDS, ETC. COMES COMPLETE WITH A HUGE COMPONENT DATABASE. SUPPORTS MONOCHROME, COLOR, OR EVEN EGA. FREE BROCHURE AND DEMO DISKETTE ON REQUEST. LIST $500 – **SPECIAL $396.15**

NEW! PC/XT EPROM PROGRAMMER $199

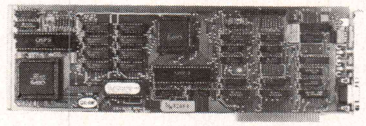


⋆ LATEST DESIGN ⋆ PROGRAMS UP TO 4 DEVICES AT ONE TIME ⋆ FEATURES EASY TO USE MENU DRIVEN SOFTWARE THAT RUNS UNDER PC OR MS-DOS. ⋆ USES AN INTELLIGENT PROGRAMMING ALGORITHM FOR SUPER FAST (8X) EPROM BURNING. ⋆ THIS PLUG-IN BOARD ATTACHES TO AN EXTERNAL MINI CHASSIS CONTAINING 4 TEXTOOL Z.I.F. SOCKETS. ⋆ NO PERSONALITY MODULES REQUIRED ⋆ AUTOMATIC VPP SELECTION: 12.5V, 21V, OR 25V. ⋆ EPROM DATA CAN ALSO BE LOADED FROM OR SAVED TO A DISKETTE. ⋆ PROGRAMMING SOFTWARE SUPPORTS: 2716, 2732, 2732A, 2764, 2764A, 27128, 27128A, 27256, 27256A, 27512, AND 27512A. ⋆ ASSEMBLED AND TESTED, BURNED. IN WITH MANUAL. $199 WITH SOFTWARE.

MICROTEK 2 M.B. EXPANDED MEMORY BOARD $195 (O. K)

MEETS LOTUS/INTEL/MICROSOFT SPEC. FOR EXPANDED MEMORY. FOR USE AS EXPANDED MEMORY (EMS), ELECTRONIC DISK, OR PRINT SPOOLER. WITH A SUPERB MANUAL AND INSTALLATION SOFTWARE. FOR PC/XT OR MOST COMPATIBLES. FOR 2 MEGABYTES OF 150 NS RAM (72 DEVICES) INSTALLED AND BURNED IN ADD $225.

PC/XT AT ENHANCED GRAPHICS ADAPTER



$249

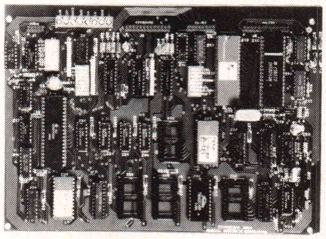
- 3 CARDS IN ONE!
- 100% IBM COMPATIBLE.
- COMPATIBLE WITH IBM EGA, COLOR GRAPHICS ADAPTER, OR MONOCHROME ADAPTER.
- 256K VIDEO RAM! (4 TIMES MORE THAN IBM!)
- DUAL FREQUENCY OUTPUT FOR EITHER EGA OR STANDARD RGB COLOR MONITORS.
- PERFECT MATE FOR NEC MULTISYNC COLOR MONITOR!
- LIGHT PEN INPUT.
- FULL 16 COLORS.
- PARALLEL PRINTER PORT.
- A SUPERIOR BOARD AT ABOUT 1/3 THE COST OF IBM EGA!
- ASSEMBLED, TESTED, AND BURNED IN. WITH MANUAL.
- USES CHIPS & TECH. VLSI 4 CHIP SET.
- FIELD PROVEN BIOS.

ZRT-80 CRT TERMINAL BOARD!

A LOW COST Z-80 BASED SINGLE BOARD THAT ONLY NEEDS AN ASCII KEYBOARD, POWER SUPPLY, AND VIDEO MONITOR TO MAKE A COMPLETE CRT TERMINAL. USE AS A COMPUTER CONSOLE, OR WITH A MODEM FOR USE WITH ANY OF THE PHONE-LINE COMPUTER SERVICES.

FEATURES:
- Uses a Z80A and 6845 CRT Controller for powerful video capabilities.
- RS232 at 16 BAUD Rates from 75 to 19,200.
- 24 x 80 standard format (60 Hz).
- Optional formats from 24 x 80 (50 Hz) to 64 lines x 96 characters (60 Hz).
- Higher density formats require up to 3 additional 2K x 8 6116 RAMS.
- Uses N.S. INS 8250 BAUD Rate Gen. and USART combo IC.
- 3 Terminal Emulation Modes which are Dip Switch selectable. These include the LSI-ADM3A, the Heath H-19, and the Beehive.
- Composite or Split Video.
- Any polarity of video or sync.
- Inverse Video Capability.
- Small Size: 6.5 x 9 inches.
- Upper & lower case with descenders.
- 7 x 9 Character Matrix.
- Requires Par. ASCII keyboard.

FOR 8 IN. SOURCE DISK (CP/M COMPATIBLE) ADD $10



$89.95 #ZRT-80 — A&T ADD $50

(COMPLETE KIT, 2K VIDEO RAM)

BLANK PCB WITH 2716 CHAR. ROM. 2732 MON. ROM **$49.95**

SOURCE DISKETTE - ADD $10

SET OF 2 CRYSTALS - ADD $7.50

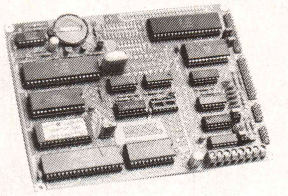
THE NEW 65/9028 VT ANSI VIDEO TERMINAL BOARD!

⋆ FROM LINGER ENTERPRISES ⋆

A second generation, low cost, high performance, mini sized, single board for making your own RS232 Video Terminal. This highly versatile board can be used as a stand alone video terminal, or without a keyboard, as a video console. VT100, VT52 Compatible.

MICRO SIZE!

FEATURES:
- Uses the new CRT9128 Video Controller driven by a 6502A CPU
- On-Screen Non-Volatile Configuration
- 10 Terminal Modes: ANSI, H19, ADM-5, WYSE 50, TVI-920, KT-7, HAZ-1500, ADDS 60, QUME-101, and Datapoint 8200
- Supports IBM PC/XT, and Parallel ASCII Keyboards
- Supports standard 15.75 kHz (Horiz.)
- Composite or Split Video (50/60 Hz)
- 25 X 80 Format with Non-Scrolling User Row
- Jump or Smooth Scroll
- RS-232 at 16 Baud Rates from 50 to 19,200
- On Board Printer Port
- Wide and Thin Line Graphics
- Normal and Reverse Attributes
- Cumulative Character Attributes: De-Inten, Reverse, Underline and Blank
- 10 Programmable Function Keys and Answerback message
- 5 X 8 Character Matrix or 7 X 9 for IBM Monitors
- Mini Size: 6.5 X 5 inches
- Low Power: 5VDC @ .7A, ± 12VDC @ 20mA.



$99.95

ADD $40 FOR A&T

(Full Kit w/100 Page Manual)

SOURCE DISKETTE: PC/XT FORMAT 5¼ IN. $15

NEW OPTION

PC/XT keyboard ROM. Allows use of IBM style keyboard. $15

Digital Research Computers

P.O. BOX 381450 • DUNCANVILLE, TX 75138 • (214) 225-2309

TERMS: Add $3.00 postage. Orders under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5-1/8% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50 add 85¢ for insurance. Prices subject to change without notice.

*TM OF DIGITAL RESEARCH INC. (CALIF.) *IBM TM OF INT. BUS. MACHINES. WE ARE NOT ASSOCIATED WITH DIGITAL RESEARCH INC. (CALIF.) THE SUPPLIERS OF CPM SOFTWARE

Circle **no. 87** on reader service card.

(continued from page 109)

time, the application program reads the menu files. Thus, you can modify the menu text with an editor, leaving the compiled application program intact. This method was chosen by the developers of the Macintosh, who call the files containing the menu data "resource files."

I'll get you started by introducing two ways to store menu information in a separate file. Each method offers a type of "menu builder" that reads a different structure for the menu text.

The first approach is the simpler one. It is based on the idea that most text editors cannot insert and display extended ASCII characters. The solution lies in the following steps:

1. Type a text that resembles the framed menu text, such that the frame characters are substituted with displayable characters (for example, the bar, hyphen, underscore, plus sign, and so on).
2. Construct a list that maps the above displayable characters with the ASCII code number of the frame characters.

Listing Five, page 99, shows a program that implements this method. The procedure *Build_Screen* reads the menu text file, which contains the mapping list discussed above (one item per line). The character set used is [!,@,#,$,^,&,/,\,¦,−,_]. The mapping list is delimited by the keyword *START*, which can be followed by two menu-frame shift coordinators. *SHIFTCOL <number>* and *SHIFTROW <number>* tell the menu builder that the menu frame is displaced by the specified number of columns and rows, respectively. The shift declarations can appear in any order and are optional, with default values of 0. The next lines contain the menu text. The menu builder scans each line, one character at a time, and translates any character found in the mapping list. Procedure *Show_Menu* displays the translated strings that form the menu, showing the sought frame characters. Procedure *DISP_STR* is used to write to the screen of an IBM PC directly for fast display.

The structure of the menu text file is simple and gives you a good idea of what the menu looks like. The sizes of the menu text files are larger than those of the next method, in which the menu text is compressed. The second technique uses a menu interpreter/builder because the menu text contains display commands. Listing Six, page 100, shows a test program using this method. The structure of the menu text file is as follows:

1. Frame parameters—Can contain any of the following directives (each is composed of a single letter followed by a number):

• *L*—Left-edge shift to specify the column position of the left edge of the menu frame. Its value ranges between 1 (default) and that of the right edge minus 4.
• *R*—Right-edge shift to specify the column position of the right edge of

DR. DOBB'S CATALOG

announcing:

TURBO PASCAL TOOLS

INCLUDING:

Turbo Advantage, a library of over 200 Turbo Pascal routines,
Turbo Display with Turbo Pascal procedures and functions to help you generate forms.
Turbo Complex to help you work with complex numbers in Turbo Pascal

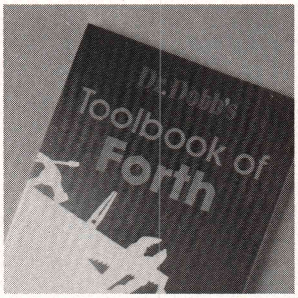
TURBO PASCAL TOOLBOOK

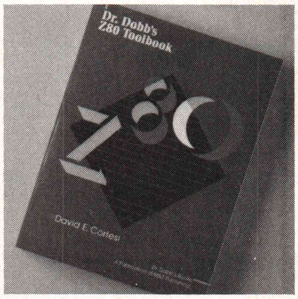
Dr. Dobb's Listings #3

including listings from Sept. through Dec. 1986

FREE! Dr. Dobb's Holiday Gift to You!

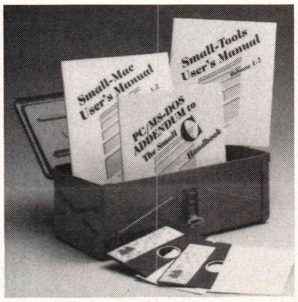
As a special holiday gift to DDJ readers, we'll send you **Dr.Dobb's Sourcebook: A Reference Guide to the C Programming Language** FREE with any order received by Dec. 31, 1986.





ALSO INSIDE:

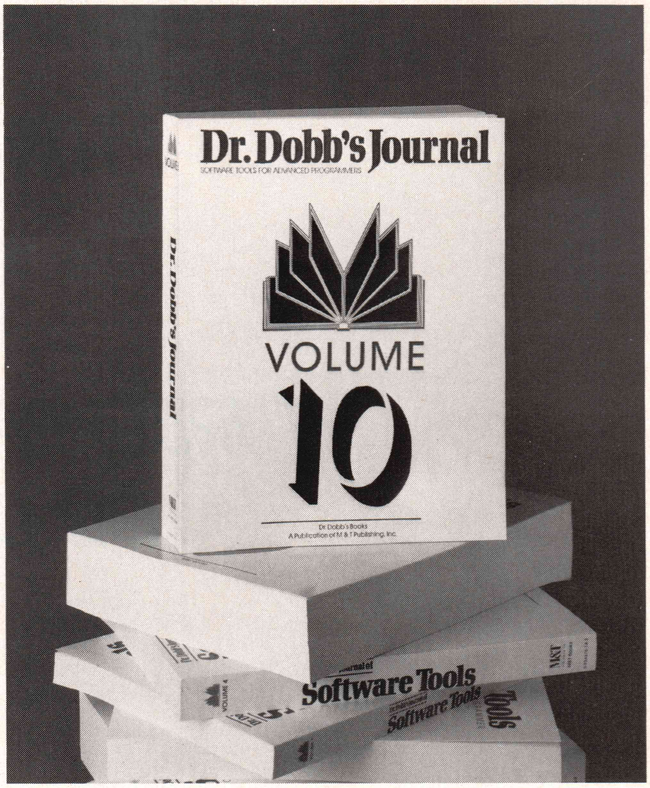
DR. DOBB'S BOUND VOLUMES
DR. DOBB'S COMPLETE C TOOLBOX
THE TOOLBOOK OF FORTH
Z80 TOOLBOOK
TOOLBOOK OF 68000 PROGRAMMING
A UNIX-LIKE SHELL & UTILITY PACKAGE FOR MS-DOS
TAMING DOS



M&T BOOKS

NEW! DR. DOBB'S BOUND VOLUME 10

Bound Volume #10: The year of living dangerously. In 1985, iconoclastic DDJ beat Apple to the goal of adding more memory, a SCSI port, and a hard disk to the Macintosh. We dared to criticize the much-praised Turbo Pascal, challenged the Unix establishment with plans for a free Unix, and asked hard questions about privacy and control in the Information Age. Of course we also kept the technical level high, with the most exhaustive review ever of programmers' editors and of C compilers, and with powerful software tools in C, Modula-2, Forth, Pascal, assembly language, and Prolog.
Item #020D $35.75



Bound Volume #1: 1976 The working notes of a technological revolution. Before there was an Apple, DDJ put a programming language on the first microcomputers, and became chronicler and instrument of the microcomputer revolution.
Item #013 $30.75

Bound Volume #2: 1977 Running light without overbyte. By year two the formula was clear: serious technical questions handled with a minimum of reverence; much source code; and a commitment to tight coding.
Item #014 $30.75

Bound Volume #3: 1978 The roots of Silicon Valley growth. The S-100 bus was hashed out in DDJ's pages. Steve Wozniak and others published in DDJ code that would help build an industry.
Item #015 $30.75

Bound Volume #4: 1979 In the midst of the gold rush. Three years before IBM moved in, the neighborhood was less civilized. DDJ published a gold mine of tips, tricks, and algorithms.
Item #016 $30.75

Bound Volume #5: 1980 C and CP/M. 1980 saw an all-CP/M issue, including Gary Kildall's history of CP/M, and Ron Cain's original Small-C compiler.
Item #017 $30.75

Bound Volume #6: 1981 The First of Forth. This was the year DDJ launched its first Forth issue and Dr. Dobb's Clinic. Plus: PCNET, the Conference Tree, and 6809 Tiny BASIC.
Item #018 $30.75

Bound Volume #7: 1982 Legitimacy. DDJ observed the IBM phenomenon, reviewed MS-DOS and CP/M-86, and looked forward to fifth-generation computers.
Item #019 $35.75

Bound Volume #8: 1983 Power tools. Professional software development on a PC was getting easier; DDJ helped, with Small-C, the RED editor, and an Ada subset.
Item #020 $35.75

Bound Volume #9: 1984 Shaping things to come. In 1984 DDJ examined new programming environments: Prolog, expert systems, Modula-2, and a $49.95 Pascal. Plus Allen Holub's GREP, Unix internals, and two encryption systems.
Item #020B $35.75

SAVE! ORDER THE COMPLETE 10 VOLUME SET!

Receive all ten Bound Volumes for only $262! You save $65!
Item #020E $262

TO ORDER: RETURN THE FORM AT THE END OF THE CATALOG, OR CALL TOLL-FREE 1-800-528-6050 EXT 4001 AND REFER TO PRODUCT ITEM NUMBER, TITLE AND DISK FORMAT

DR. DOBB'S 1986 LISTINGS ON DISK

Dr. Dobb's Journal of Software Tools offers the convenience of selected listings on disk.

DR. DOBB'S LISTINGS #1

JANUARY-APRIL 1986

Includes listings from the following articles, and more.

January Issue #111

"A Simple OS for Realtime Applications; 68000 Assembly Language Techniques for an Operating System Kernel" by *DDJ* editor Nick Turner.

"32-bit Square Roots; An 8086 Assembly Language Routine for 32-bit Square Roots" by Michael Barr.

February Issue #112

"Data Abstraction with Modula-2" by Bill Walker and Stephen Alexander.

"Learning Ada on a Micro; A Draw Poker Program in Ada" by DoWhile Jones.

March Issue #113

"A Variable-Metric Minimizer; A C Program for Minimizing Arbitrary Functions" by Joe Marasco.

"Concurrency and Turbo Pascal; An Approach to Implementing Coroutines in Pascal" by Ernest Bergmann.

April Issue #114

"Boca Raton Inference Engine; Lisp, Prolog, and Expert-2 Techniques and Code" by Robert Brown.

Dr. Dobb's Listings #1/86 Item #170 $14.95

DR. DOBB'S LISTINGS #2

MAY-AUGUST 1986

Includes listings from the following articles, and more.

May Issue #115

"Simple Plots with the Enhanced Graphics Adapter" by Nabajyoti Barkakati.

"The Cryptographer's Toolbox" by Fred A. Scacchitti.

June Issue #116

"Structured Programming; Overloading Procedures, Exporting Opaque Types, Data Hiding" by Namir Shammas.

"Compuserve B Protocol" by Steve Wilhite.

July Issue #117

"Structured Programming; Tiny Tools, Array-Defining Words" by Michael Ham.

August Issue #118

"Structured Programming; Generic Routines in Ada and Modula-2, Extended For Loop" by Namir Shammas.

Dr. Dobb's Listings #2/86 Item #171 $14.95

Please specify MS-DOS, Macintosh, or CP/M.
For CP/M disks specify: Apple, Osborne, Kaypro, Zenith

Dr. Dobb's Listings #3

September - December 1986

Includes listings from the following articles, and more:

September Issue #119

"Algorithms: Curve Fitting with Cubic Splines" by Ian E. Ashdown "C Chest: Directory Traversal, Trailing Zs, and Horrifying Experiences" by Allen Holub

October Issue #120

"C Chest: More, a File-Browsing Utililty" by Allen Holub

"Processors: TNZ: An 8-bit to 16-bit Translator" by Richard Campbell

November Issue #121

"Digital Dissolve" by Mike Morton

"Mandelbrot Set" by Howard Katz

December Issue #122

"In Search of a Sine" by R.A. Campbell

16-bit Software Toolbox: A demo program to illustrate the undocumented MS-DOS Int 2EH route to the command interpreter in command.com by Ray Duncan and his readers, and 8086-68000 based string comparison routines.

Dr. Dobb's Listings #3/86 Item #172 $14.95

TO ORDER: RETURN THE FORM AT THE END OF THE CATALOG, OR
CALL TOLL-FREE 1-800-528-6050 EXT 4001
AND REFER TO PRODUCT ITEM NUMBER, TITLE AND DISK FORMAT

DR. DOBB'S C TOOLBOX

DR. DOBB'S TOOLBOOK OF C

Over 700 pages of C material, including articles by such C experts as Kernighan and Ritchie, Cain and Hendrix, Skjellum and Holub! The level is sophisticated and pragmatic. The most valuable part of the *Toolbook* to many will be the hundreds of pages of useful C source code, including: Jim Hendrix's famous Small-C Compiler and New Library for Small C—Also available on disk!; NEW! Hendrix's Small Mac: An Assembler for Small C and Small Tools: Programs for Text Processing—Both also available on disk!; and all of Anthony Skjellum's C Programmer's Notebook columns distilled by Tony into one thought-provoking chapter.

From M&T Publishing and Brady Communications

Dr. Dobb's Toolbook of C **Item #005** **$29.95**

SMALL-C COMPILER

Jim Hendrix's Small C Compiler is the most popular piece of software published in *Dr. Dobb's* 11-year history. Like a home-study course in compiler design, the Small-C Compiler and the *Small-C Handbook* provide everything you need but the computer for learning how compilers are constructed, and for learning C at its most fundamental level.

Small-C Compiler **Item #007** **$19.95**

SMALL-MAC:
AN ASSEMBLER FOR SMALL-C

This assembler features simplicity, portability, adaptability, and educational value. The package includes: a simplified macro facility; C language expression operators; object file visibility; descriptive error messages; and an externally defined instruction table.

You get the macro assembler, linkage editor, load-and-go loader, library manager, CPU configuration utility, and a utility to dump relocatable files. Documentation is also included.

For CP/M systems only. Please specify format.

Small-Mac **Item #012A** **$29.95**

SMALL-TOOLS:
PROGRAMS FOR TEXT PROCESSING

This package of programs performs specific, modular operations on text files, including: editing; formatting; sorting; merging; listing; printing; searching; changing; transliterating; copying; concatenating; encrypting and decrypting; replacing spaces with tabs and tabs with spaces; counting characters, words, or lines; and selecting printer fonts.

Small-Tools is supplied in source code form only. With the Small-C Compiler you can select and adapt these tools to meet your own needs. Documentation is included.

Small-Tools **Item #010A** **$29.95**

THE SMALL-C HANDBOOK

Jim Hendrix's *Small-C Handbook* is the reference book on his Small-C Compiler. In addition to describing the operation of the compiler, the book contains complete source listings to the compiler and its library of arithmetic and logical routines.

A perfect companion to the Hendrix Small-C Compiler available from *DDJ* on disk, the *Handbook* even tells you how to use the compiler to generate a new version of itself!

While both the *Handbook* and the *Toolbook* provide documentation for the Small-C Compiler, the *Handbook* contains a more detailed discussion and is available with addendum for the MS/PC-DOS version.

From M&T Publishing and Brady Communications

The Small-C Handbook **Item #006** **$17.95**

The Handbook with MS/PC-DOS Addendum

Item #006A **$22.95**

C DISK FORMATS

When ordering, please indicate MS/PC DOS or CP/M. For CP/M disks, please specify one of the follwing formats: Apple, Osborne, Kaypro, Zenith Z-100 DS/DD, 8" SS/SD. Special order formats are available for an additional $10 each.

SPECIAL PACKAGES 20% OFF

CP/M C PACKAGE

Receive this special package and save $20!

You'll get: Dr. Dobb's Toolbook for C; The Small-C Handbook; The Small-C Compiler on disk; The Small-Mac assembler on disk, with documentation; and The Small-Tools text-processing programs on disk, with documentation all for only $99.95!

Please specify format.

CP/M C Package **Item #005A** **$99.95**

MS/PC-DOS C PACKAGE

Save $20 when you order this special package.

You'll receive: Dr. Dobb's Toolbook of C; The Small-C Handbook with the MS/PC DOS Addendum; The Small-C Compiler on disk; and The Small Tools text-processing programs on disk, with manual all for only $82.95.

MS/PC-DOS C Package **Item #005B** **$82.95**

TO ORDER: RETURN THE FORM AT THE END OF THE CATALOG, OR CALL TOLL-FREE 1-800-528-6050 EXT 4001 AND REFER TO PRODUCT ITEM NUMBER, TITLE AND DISK FORMAT

SH: A UNIX-LIKE SHELL FOR MS-DOS

Includes complete source code and documentation

THE SHELL V.2
NEW, UPGRADED VERSION!

The shell is an MS-DOS implementation of the most often used parts of the UNIX C Shell. This package includes an executable version of the Shell, along with **complete source code and full documentation**, by DDJ columnist Allen Holub.
If you are a registered user, or have already purchased Version 1 of the Shell from Dr. Dobb's, you can receive the upgrade disk for only $6. Supported features include:

Editing Command-line editing with the cursors is supported. The line is visible as you edit it.
Aliases Can be used to change the names of commands or as very fast, memory-resident, batch files. Nested aliases are supported.
History You can execute previous commands. The command can be edited before being executed. Version 2 supports imbedded history requests (Bar; !! >foo).
Redirection and Pipes <> >> >& >>& |
Pipe temporary files can be put on a RAM disk.
Unix-like Command Syntax / can be used to separate directory names (\ can now be used as well). A 2048-byte commnd line is supported. Command-line wild card expansion. Multiple commands on a line.
DOS-compatible prompt support
$d $t $_ $e $h $n $q $$ $%
C-Shell Based Shell Scripts (batch files) Shell Variables are macros that can be used on the command line. Version 2 supports arithmetic manipulation of shell variables using the @ command. The following C operators are supported: () + − * / % <= >= <> != == ! && || =
A batch file can call another batch file like a subroutine. Control is passed to the second file and then back to the first when the second is finished. Batch files can return values to the calling file using the exit and $status mechanisms.

A powerful, interpretive, programming language, based on the UNIX C Shell, is now supported, including:

if/then/else	foreach	break
while	switch/case	continue

All commands can be nested.
The shell runs on IBM PC's and compatibles.

The Shell	**Item #160**	**$29.95**
The Shell Upgrade Disk *(for owners of the Shell Version 1)*	**Item #160A**	**$6.00**

SAVE OVER 15%! ORDER THE SHELL AND /UTIL TOGETHER FOR ONLY $50!

Shell /Util Package	**Item #162**	**$50**

/UTIL

/Util is a collection of UNIX-like utility programs for MS-DOS. This package includes updates of the highly acclaimed Dr. Dobb's articles; Grep: a UNIX-like Generalized Regular Expression Processor, and LS and Getargs from DDJ's C Chest.
Source code is included and all programs (and most of the utility subroutines) are fully documented in a UNIX-style manual. You'll find executable versions of:

cat	echo	mv	rm
cp	grep	p	rmdir
date	Ls	pause	sub
du	mkdir	printenv	chmod

/Util	**Item #161**	**$29.95**

TO ORDER: RETURN THE FORM AT THE END OF THE CATALOG, OR CALL TOLL-FREE 1-800-528-6050 EXT 4001 AND REFER TO PRODUCT ITEM NUMBER, TITLE AND DISK FORMAT

THE DR. DOBB'S TOOLBOOK SHELF

Z80 TOOLBOOK

David E. Cortesi, longtime Dr. Dobb's columnist brings you:

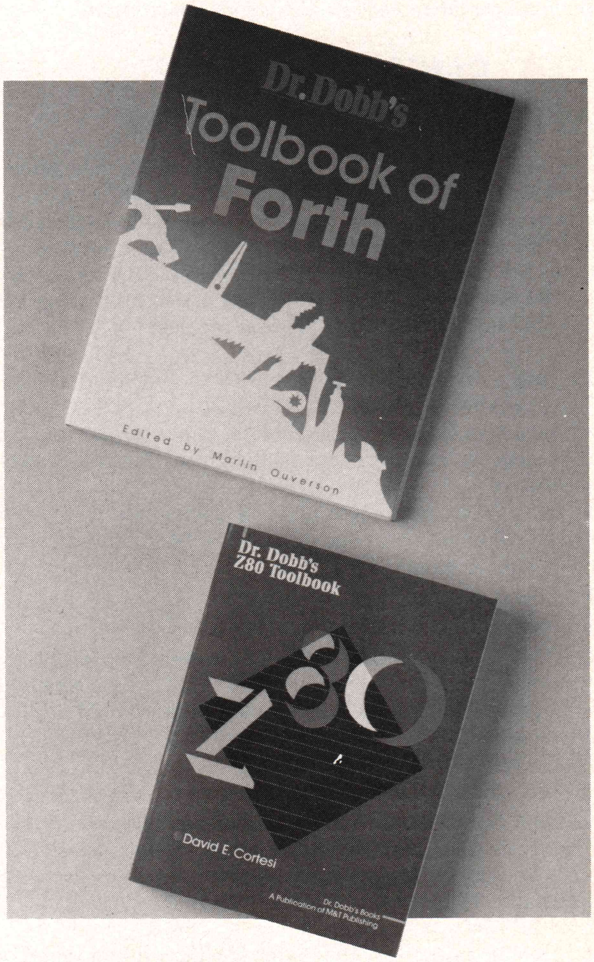
- **a method of designing programs and coding them in assembly language.** Cortesi walks through the initial specifications, designing an algorithm and writing the code. He demonstrates the construction of several useful programs.
- **a complete, integrated toolkit of subroutines** for arithmetic, for string-handling, and for total control of the CP/M file system. They bring the ease and power of a compiler's run-time library to your assembly language work, without a compiler's size and sluggish code.
- **Every line of the toolkit's source code is there to read.**

ORDER THE Z80 SOFTWARE ON DISK!

All the software in Dr. Dobb's Z80 Toolbook—the programs plus the entire toolkit, both as source code and object modules for both CP/M 2.2 and CP/M Plus—is yours on disk! Most of the programs are included in the book, however, the disk is necessary for complete listings. A Z80 microprocessor and a Digital Research International RMAC assembler or equivalent are required.

Dr. Dobb's Z80 Toolbook	**Item #022**	**$25**
Dr. Dobb's Z80 Toolbook w/disk	**Item #022A**	**$40**

Please specify one of the following disk formats: 8" SS/SD, Apple, Osborne, or Kaypro



DR. DOBB'S TOOLBOOK OF FORTH

This comprehensive collection of useful Forth programs and tutorials contains DDJ's best Forth articles, expanded and revised along with new material. In addition, you'll glean important insights about the potential of this increasingly popular language from the many in-depth discussions of advanced Forth topics. You'll find sections on:

Mathematics in Forth, including "Series Expansion in Forth," "Forth Floating-Point Package," and "Signed Integer Division"
Modifications/Extensions, including "A Proposal for Strings in Forth," "Non-Deterministic Control Words," "Some Forth Coding Standards," and "Towards a More Writable Forth Syntax"
Forth Programs, including "GO in Forth," "Elements of a Forth Data-Base Design," "The Forth Sort," "SEND & RECV," "Interface for a Mouse," "Relocating Loader in Forth," "Forth Decompiler," "Screen-Oriented Editor ReVisited," "Evolution of a Video Editor," "H-19 Screen Editor," and "The Conference Tree."
Forth—the language, including "The Forth Philosophy," "Teaching Forth as a First Language," and "Forth-83 and Vocabularies"
Implementing Forth, including "Forth and the Motorola 68000," "A 68000 Forth Assembler," "A Forth Assembler for the 6502," and "Z8000 Forth." You'll also find Appendices that will help you convert fig-Forth to Forth-83, and tell you how to stay up-to-date on the latest developments and refinements of this popular language.

The screens in the book are also available on disk as ASCII files. Receive Dr. Dobb's Toolbook of Forth, along with the software on disk, together for only $39.95.

Dr. Dobb's Toolbook of Forth	**Item #030**	**$22.95**
Dr. Dobb's Toolbook of Forth w/Disk	**Item #031**	**$39.95**

Please specify MS/PC-DOS, Apple II, Macintosh, or CP/M. For CP/M disks, specify Osborne or 8" SS/SD.

TO ORDER: RETURN THE FORM AT THE END OF THE CATALOG, OR
CALL TOLL-FREE 1-800-528-6050 EXT 4001
AND REFER TO PRODUCT ITEM NUMBER, TITLE AND DISK FORMAT

DR. DOBB'S LATEST RELEASES

DR. DOBB'S TOOLBOOK OF 68000 PROGRAMMING

In this complete collection of practical programming tips and techniques for the 68000 family, you'll find the best articles on 68000 programming ever published in Dr. Dobb's, along with new material from 68000 experts. You'll learn about the most important features of the 68000 microprocessor from a full description and concise discussion of its history and design. And, useful applications and examples will show you why computers using the 68000 family are easy to design, produce and upgrade. **Contents include:**

an Introduction to the 68000 Family
- 68000 Instruction Set

Development Tools
- Bringing Up the 68000: A First Step
- A 68000 Cross-Assembler

Useful 68000 Routines and Techniques
- A Simple Multitasking Kernel for Real-Time Applications
- The Worm Memory Test
- A Mandelbrot Program for the Macintosh

All programs are available on disk!

In addition, an executable version of the 68000 Cross-Assembler can be purchased along with the source code and documentation for $25. (Requires one or more disk drives and either CP/M-80, CP/M 2.2 with 64 bytes or MS-DOS with 128 bytes)

Dr. Dobb's 68000 Programming Toolbook	**Item #040**	**$29.95**
68000 Toolbook with Disk	**Item #041**	**$49.95**

Please specify one of the following disk formats: CP/M 8″, Osborne, Macintosh, Amiga, Atari 520st, MS-DOS

68000 Cross-Assembler	**Item #042**	**$25**

STAT TOOLBOX FOR TURBO PASCAL

The STAT TOOLBOX is two statistical packages in one! Whether you're a programmer looking for tools to build your own applications, or a user who wants a complete, fully functioning package, the STAT TOOLBOX will bring convenience, power and versatility to your statistics programs!

The STAT TOOLBOX is written in Turbo Pascal and includes full source code. It contains two complete packages:

A reference disk and manual

Flexible, time-saving building blocks allow you to customize statistical applications to fit your needs. You'll find:
- statistical distribution functions;
- random number generation;
- basic descriptive statistics
- parametric and non-parametric statistical testing.
- bivariate linear regression, multiple and polynomial regression
- automatic best curve selection

A demonstration disk and manual

The demonstration package includes fully functioning statistical programs, and two data management programs. In addition to these practical tools, files containing sample data will help users learn to enter, edit, maintain and manipulate data.

High resolution graphics capabilites include drawing histograms and regression lines, and plotting residuals for regression analysis. (Borland's Turbo Graphic's Toolbox is required)

For IBM PC's and compatibles. Turbo Pascal version 2.0 or later, and PC-DOS 2.0 or later are required.

Stat Toolbox	**Item #050**	**$69.95**

TAMING DOS BY THOM HOGAN

Learn how to make DOS work for YOU! **Taming DOS** will take you beyond the basics, picking up where your DOS manual leaves off. You'll find batch files and DOS enhancements that extend the power of DOS so you can work more accurately and efficiently. And, you'll learn how to customize DOS to fit YOUR needs, saving you time and frustration every time you use it.

Taming DOS will show you how to create a memory-resident clock, rename subdirectories and change file attributes. You'll learn more about what's on your disk and how to protect it easily, how to create configurable AUTOEXEC.BAT files and understand redirection and piping. **You'll find:**
- how to customize CONFIG.SYS and use ANSI.SYS to change the appearance of DOS
- extensive batch file coverage
- example routines using redirection, including redirection operators, filters, and pipes
- DOS enhancement programs that are ready to use, with executable commands.
- Full source code, allowing you to alter programs and create a custom system.

The book includes assembly language programs that allow you to manipulate DOS at its lowest form. An assembler is not needed to enter the programs; **Taming DOS** presents an alternative method for anyone with BASIC on their machine. The programs, including batch files and DOS enhancements are also available on disk along with source code.

Taming DOS	**Item #060**	**$19.95**
Taming DOS with disk	**Item #061**	**$34.95**

TO ORDER: RETURN THE FORM AT THE END OF THE CATALOG, OR
CALL TOLL-FREE 1-800-528-6050 EXT 4001
AND REFER TO PRODUCT ITEM NUMBER, TITLE AND DISK FORMAT

TURBO PASCAL TOOLS

TURBO ADVANTAGE: SOURCE CODE LIBRARIES FOR TURBO PASCAL

Here's the advantage you need to make your programming easier and more efficient! This library of 220 routines, complete with source code, sample programs and documentation, will save you hours developing and optimizing your programs. A few of the helpful **Turbo Advantage** files include:
* arithmetic operations * bit manipulations * check routines * data compression * differentiation and integration * Fourier analysis and synthesis * file management and hash methods * Input/Output routines * matrix processing * menu functions * MS-DOS support * linear and nonlinear optimization * screen/printer redefinition * sorting routines * spline integration, differentiation, interpolation * statistical procedures * string processing * type conversion
All procedures and functions are supplied in source code form so you can adapt them to meet your needs. Each file contains a list of relevant routines, all explained with example programs to help you understand the callings, parameters and data types in detail. Most of the programs can be immediately compiled. The detailed manual includes a short characterization of the routine, a description of the way the routine works with an explanation of the methods used, the calling sequence—including a description of the parameters and the function result, notes with advice and special explanations, and a simple example illustrating the files and variables needed.

Turbo Advantage is for MS/PC DOS systems.
Item #070 $49.95

TURBO COMPLEX ADVANTAGE: COMPLEX NUMBER ROUTINES FOR TURBO PASCAL

Working with complex numbers, vectors and matrices is easy with **Turbo Advantage Complex!** This library of over 80 procedures and routines will help you program complicated functions more easily. You'll find routines to help you:

* use digital filters, solve boundary-value problems and process very large arrays;
* carry out vector and matrix calculations with complex integers and variables;
* execute time-saving simultaneous Fourier transforms and calculations of convolution and correlation functions.

The **Turbo Complex** package includes source code and documentation. For MS-DOS systems. Some of the Turbo Complex routines are most effectively used with routines contained in the Turbo Advantage package.
Item #071 $89.95

THE TURBO PASCAL TOOLBOOK

The Turbo Pascal Toolbook contains routines and sample programs to make your programming easier and more powerful.

• **Mathematical Expression Parsers** offers two routines that convert mathematical expressions into RPN tokens.

• **External Sorting and Searching Tools** presents a new database routine that combines the best features of the B-tree. B+ and B++ trees.

• **When Turbo Isn't Enough** presents an extensive library of low-level routines that show you how to tap into the hardware and the operating system.

• **Window Management** will help you create, sort and overlay windows.

• **A Smart Statistical Regression Model** automatically searches for the best regression model to represent a given set of data. The routine libraries and the sample programs in The Turbo Pascal Toolbook are also available on disk for MS-DOS systems. Complete source code is included.

The Turbo Pascal Toolbook **Item #081 $25.95**
The Turbo Pascal Toolbook with disk **Item #081 $45.95**

TURBO DISPLAY ADVANTAGE: FORM GENERATOR FOR TURBO PASCAL

Now, even if you have little programming knowledge, you can design and process forms to fit your needs! The Turbo Display form generator includes a text editor, 30 time-saving Turbo Pascal procedures and functions, and a concise, informative handbook to help you take full advantage of all Turbo Display capabilities.

Turbo Display includes source code and documentation. For MS-DOS systems. Some of the Turbo Advantage routines are necessary to compile Turbo Display.
Item #072 $69.95

ORDER FORM

ORDER NOW!

NAME ______________________________
(Please use street address, not P.O. Box)
ADDRESS ______________________________
CITY ______________ STATE ________ ZIP ________
DAY PHONE ______________________________

For disk orders, please indicate format. Refer to ad for standard format availability for each product. Special formats available for additional $10 each.

☐ MS/DOS ☐ CP/M ___ Zenith Z-100 DS/DD
☐ Macintosh ___ Kaypro ___ Osborne ___ Amiga
☐ Apple II ___ 8″ SS/SD ___ Apple ___ Atari 520st

For Faster Service On Credit Card Orders

CALL TOLL FREE 1-800-528-6050 Ext. 4001

Please Refer To Item # When Ordering

MasterCard VISA AMERICAN EXPRESS

QUANTITY	ITEM #	DESCRIPTION	UNIT PRICE	TOTAL PRICE

CA residents must add applicable sale tax on merchandise total ________ % → **SALES TAX**
(CA residents must add sales tax EXCEPT *Dr. Dobb's Sourcebook* #004)
Shipping must be included with order. See rates below. → **SHIPPING**

SUB-TOTAL ________
SALES TAX ________
SHIPPING ________
TOTAL ORDER ________

☐ VISA — NAME ON CARD ______________________________
☐ MASTERCARD — ACCOUNT NO. ______________________________
☐ AMERICAN EXPRESS — EXPIRATION DATE ______________________________
☐ CHECK (make checks payable to M&T Publishing) — SIGNATURE ______________________________

In U.S. For Bound Volumes, add $3.25 per book. Add $8.75 for special C Packages. For other books and disks, add $2.25 per item.
Outside U.S. For Bound Volumes, add $6.25 per book surface mail. Add $18 surface mail for Special C Packages. For other books and disks, add $5.25 per item surface mail. Foreign airmail rates available on request.

For Faster Service and reduced shipping costs, Europeans may order direct from: Markt & Technik, Buchverlag, Hans-Pinsel-Strasse 2, 8013 Haar bei München. Call Germany 89-4613-221 for prices in Deutch Marks.

PROMPT DELIVERY! DEALER INQUIRY WELCOME!

DR. DOBB'S CATALOG ORDER FORM

Please fold along fold-line and staple or tape closed.

No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL

First Class   Permit No. 790   Redwood City, CA

Postage Will Be Paid By Addressee

DR.DOBB'S CATALOG

501 GALVESTON DRIVE
REDWOOD CITY, CA 94063

Please fold along fold-line and staple or tape closed.

(continued from page 112)

the menu frame. Its value ranges between that of the left edge plus 4 and 80 (the default).

• *H*—Specifies either 1 or 2 as the number of lines in the horizontal menu-frame edges.

• *V*—Specifies either 1 or 2 as the number of lines in the vertical menu-frame edges.

2. *START* delimiter—Used because the above parameters are optional and can appear in any sequence; it is used to delimit this set of parameters from the optional column and row shift parameters.

3. Column/row shift parameters—Use the keywords *SHIFTCOL* or *SHIFTROW*, followed by the corresponding magnitude. Give the column and row distance between the upper-left corners of both the physical screen and the menu frame.

4. Menu-building commands—Provide the following symbols and characters used in building the desired menu and its frame:

• *@<3-char code>*—Used to specify a single extended ASCII-coded character. Function *Get_Char_Code* documents the three-letter codes of various parts of the menu frame. For example, *ULC* represents the upper-left corner.

• *D<3-char code> <2-digit repeat factor>*—Used to duplicate a frame character. The three-character codes are used to specify the frame character, and two subsequent digits specify the number of times the character is duplicated.

• *S<2-digit>*—Used to skip the number of columns specified by the two digits.

• Double quote—Used to signal that a menu text follows. A bar symbol is used optionally to delimit the text.

• *#<2-digit>*—Used to jump to the column position specified by the two-digit number.

• *V*—Used to draw the left and right vertical frame edges with spaces in between.

• *H*—Used to draw a horizontal line between the left and right edges.

The procedures *DISP_STR* and *Show_Menu* are used as in the first method to display the menu in question quickly.

Tables 1 and 2, below, show menu text files that display the same framed menu using the first and second methods, respectively. The text in Table 2 is terser and more cryptic and thus is recommended for frequent use. For casual users, I recommend the first method.

Because the two methods are not language or hardware dependent, they can be implemented easily in other languages. For programmers who work with multiple languages and translate programs, this approach is attractive because the text files are reused without any change. With menu-intensive applications it is more feasible to translate the menu builders.

Bibliography

Beidler, J., and Jackowitz, P. *Modula-2*. Boston, Mass.: PWS Publishers, 1986.

DDJ

(Listings begin on page 96.)


```
/201    <-- Upper-left corner
\187    <-- Upper-right corner
&200    <-- Lower-left corner
%188    <-- Lower-right corner
$204    <-- Left tee
@185    <-- Right tee
^202    <-- Upper tee
!203    <-- Lower tee
-205    <-- Horizontal line
¦186    <-- Vertical line
START

/---------------------------------------------!------------------\
¦                                             ¦                  ¦
¦   M  A  I  N        M  E  N  U              ¦      [Q]uit      ¦
¦                                             ¦     [Esc]ape     ¦
$---------------------------------------------^ -----------------@
¦  1) Prepare data                                               ¦
¦                                                                ¦
¦  2) Calculate results                                          ¦
¦                                                                ¦
¦  3) Print results                                              ¦
¦                                                                ¦
&----------------------------------------------------------------%
```

Table 1: *Contents of a menu text file using the first menu-creation method. Comments have been added to the coded symbols.*

```
H2
V2
L01
R56
START
@ULC DHLN36 @DNT DHLN17 @URC
@VLN S36 @VLN S17 @VLN
@VLN S08 "M A I N M E N U¦ S09 @VLN " [Q]uit¦ S08 @VLN
@VLN S36 @VLN " [Esc]ape ¦ @VLN
@VLN DHLN36 @UPT DHLN17 @VLN
@VLN " 1) Prepare data¦ #54 @VLN
V
@VLN " 3) Calculate results¦ #54 @VLN
V
@VLN " 3) Print results¦ #54 @VLN
@LLC DHLN54 @LRC
```

Table 2: *Contents of a menu text file using the second method. The menu displayed is equivalent to that obtained in Table 1.*

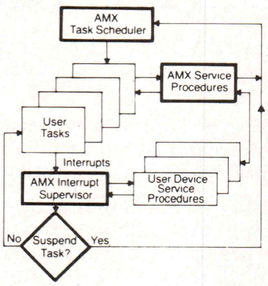
8080
Z80
6809

AMX

8086
8088
Now
68000

Real-Time Multitasking Executive

- No royalties
- Source code included
- Fault free operation
- Ideal for process control
- Timing control provided
- Low interrupt overhead
- Inter-task messages



Options:

- Resource Manager
- Buffer Manager
- Integer Math Library

AMX is TM of KADAK Products Ltd.
CP/M-80 is TM of Digital Research Corp.
IBM, PC DOS are TM of IBM Corp.

- Language Interfaces :
  C Pascal
  PL/M Fortran
- DOS File Access :
  CP/M-80
  IBM PC DOS

AMX for	8080	$ 800 US
	8086	950
	6809	950
	68000	1600
Manual (specify processor)		75

KADAK Products Ltd.
(604) 734-2796
Telex: 04-55670
206–1847 W. Broadway, Vancouver, B.C., Canada V6J 1Y5

Circle **no. 325** on reader service card.

C and Assembly Language Programmers!!

asmTREE (tm)

- **B+tree relational database development system for C and Assembly Language environments under MS-DOS.**

- Multiple keys per data file
- Temporary indices may be created at run time.
- Wildcard key value searches
- ASCII, binary, and floating point key values.
- Virtual file handling allows any number of data and index files to be used.
- Source code is available.
- No royalties req'd for your application programs.

. . . and it's FAST!! Written 100% in Assembly Language for lightning fast execution and small code size.

ASMLIB (tm)

Assembly language development LIBRARY with over 200 functions which give you - Text Windowing, Floating Point Math and Trig., High level graphics, etc. . . .

asmTREE .... only $395.00 COMPLETE
ASMLIB .... only $149.00 COMPLETE

Requires IBM PC/XT/AT or compatible running under MS-DOS or equivalent.

BC Associates
3261 N. Harbor, Suite B
Fullerton, CA 92635

(714) 526-5151

Circle **no. 182** on reader service card.

C Bricklin Run

C-scape: A Prototyping Tool

"Everybody who writes software, either commercially or for in-house applications, should immediately order a copy of ***Dan Bricklin's Demo Program.*** *Period. No exceptions."* Soft·letter

C-scape should immediately be included in that quote. Use Dan Bricklin's Demo Program to design exactly how each screen should appear. You can even use Demo to capture existing screens, making language conversion projects easier. Then use **C-scape's demo2c** utility to turn each screen into C code, complete with text, menus, input fields, masking, and colors. Screens will compile the first time. Then link in **C-scape's library** functions for scrolling, validation, type support, and much, much more, and presto—your program is done.

C-scape: A Screen Generator and Library

"Dan Bricklin's Demo Program without **C-scape** *is like VisiCalc without a printing function."* Joseph Katz, C Notes, Sextant Magazine

C-scape is by far the most flexible and easiest to learn and use screen library available. It offers over 100 useful functions, and its unique design makes it easy to create your own functions with maximum support and minimum constraint. It's a veritable function construction kit. Combined with Dan Bricklin's Demo Program, **C-scape** will accelerate your programming productivity, boost your creativity, reduce coding errors, and facilitate maintenance. No more screens from scratch. No more procrustean screen libraries.

Fully definable validation • full type support • automatic scrolling • **full *printf* % substitution inside field definitions** • Lotus-like/pop-up/pull-down menus • **menus, data entry fields, windows, prompts, text** • supports standard C types • full color support • **completely definable keys** • full screen I/O support • user-definable field types • **source code included** • input masking • field marking • orthogonal and direct field movement • civilized, easy-to-remember function names • modular construction makes debugging/maintenance easy • generic data pointer makes customization easy • **time and money functions** • 32,000 + fields • error checking aids program development • clear and concise documentation • **800 support and bulletin board.**

30-day guarantee: Following registration you'll get full source code. No run-time license, no royalties.
$149.00 C-scape (Lattice 3 + /Microsoft 3 + ; others call) **$219.00 C-scape with Dan Bricklin's Demo Program**
Please add $3.00 for shipping. Massachusetts orders please add 5% sales tax.

Another Key to Freedom, from
Oakland Group, Inc.
675 Massachusetts Avenue, Cambridge, MA 02139

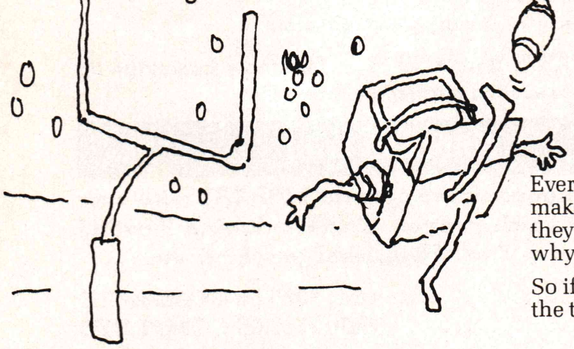
MasterCard VISA

Call today!
617-491-7311 or
800-233-3733

Circle **no. 227** on reader service card.

Prices constantly change!
Call for latest low prices.

Introducing the PC Designs All-American Team.

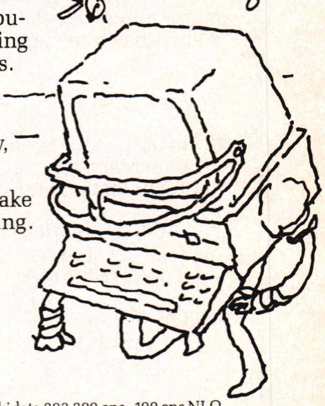


Here in Tulsa, we know a thing or two about building a winning team. And the first rule is: Champions aren't born, they're made.

At PC Designs, that same approach has earned us a reputation for quality, performance and reliability that's making the "Limited" team in Longhorn country mighty envious.

Every peripheral listed below has been thoroughly tested to make sure it lives up to our high standards. And, of course, they all live up to our low prices. As for our speed of delivery, why do you think they call Oklahoma the "Sooners?"

So if you haven't talked to the folks here at PC Designs yet, make the two-point conversion: 1) Give us a call; 2) Order something.



A Note About Prices . . .

You may well find lower advertised prices elsewhere. But before you buy, ask the following questions:

1) Is the item available?
2) Do they offer service?
3) Is the item first quality (some mail-order outfits sell hard disks that are "seconds" or discontinued models).
4) Does the price of a hard drive include cable and a high quality controller?
5) Is shipping included in cost?

The list goes on . . . call PC Designs today for more tips on buying **smart**.

ALL PRICES INCLUDE SHIPPING!

Hard Disk Drives

All hard disks include:
- detailed installation and operations manual
- formatting software
- low level format in place
- automatic head park on power down
- all necessary cabling and mounting hardware/rails (AT)
- boot from hard drive
- one year warranty

XT Hard Drives (include hard disk controller):

20 megabyte Seagate ½ height ST225 (65MS Access Time)
Price $399

30 megabyte Seagate ST238 ½ height (65 MS Access Time)
Price $489
(includes Adaptec RLL controller)

AT Hard Drives (Linear Voice Coil Activator):

Seagate:

20 megabyte Seagate ST4026 (39MS Access Time)
Price $545

30 megabyte Seagate ST4038 (39 MS Access Time)
Price $629

40 megabyte Seagate ST4051 (39 MS Access Time)
Price $745

80 megabyte Seagate ST4096** (25 MS Access Time)
Price $1,275

**Ask about our BIOS Customization for high capacity hard drives!

CORE:

20 megabyte ATPlus (28 MS Access Time)
Price $1,395

32 megabyte ATPlus (28 MS Access Time)
Price $1,495

40 megabyte ATPlus (28 MS Access Time)
Price $1,595

56 megabyte ATPlus (28 MS Access Time)
Price $2,495

72 megabyte ATPlus** (28 MS Access Time)
Price $2,995

**Ask about our BIOS Customization for high capacity hard drives!

Priam:

24 megabyte (28 MS Access Time)
Price .... Call

40 megabyte ID-40 AT (28 MS Access Time)
Price .... Call

60 megabyte ID-60 AT (28 MS Access Time)
Price .... Call

Streaming Tape Backup Units

All tape units include:
- detailed installation and operations manual
- state-of-the-art menu or command driven, performance minded streaming tape software
- industry standard controller interface card
- high speed backup—5 megabytes per minute
- mounting hardware/rails and cables

Everex Excel Stream 20 Internal (Cassette) $589
Everex Excel Stream 20 External (Cassette) $669
Everex Excel Stream 60 Internal (¼ inch Cartridge) $759
Everex Excel Stream 60 External (¼ inch Cartridge) $829

Modems

Includes:
- telecommunications software
- fully Hayes command set compatible
- low power, internal ½ size cards

Evercom 300/1200 baud $129
Evercom 2400 baud internal $289
Astec 300/1200 baud $129

Monitors

- High Resolution Amber Monochrome $95
- Princeton Graphics Max-12 .... $179
- Princeton Graphics HX-12 RGB $459
- Princeton Graphics HX-12E (EGA Compatible) $539
- NEC Multisync (CGA, EGA and PGA Compatible) .... Call
- Taxan 760 14" EGA Compatible) $629

Video Display Boards

- Monochrome Graphics Compatible 720 x 350 .... $100
- Everex Edge Multiple Display Card $249
- Everex Graphics Edge $249
- EGA Compatible .... Call

Printers

Because of the nature of our agreement with some of the manufacturers we are not allowed to advertise our low prices. We are able to offer excellent prices—please call.

Dot Matrix:
- Epson LQ-800 24 pin, 180 cps, 60 cps NLQ, 80 col .... save
- Epson LQ-1000 24 pin, 180 cps, 60 cps NLQ, 136 col .... save
- Epson FX-85 160 cps, 32 cps .... save
- Epson FX-286 200 cps, 40 cps NLQ .... save
- Epson LQ-1500 200 cps, 67 cps NLQ .... save
- Okidata 182 120 cps, 60 cps NLQ, 80 col .... save
- Okidata 183 120 cps, 60 cps NLQ, 136 col .... call
- Okidata 192 160 cps, 80 cps NLQ, 80 col .... call
- Okidata 193 160 cps, 80 cps NLQ, 136 col .... call
- Okidata 292 200 cps, 100 cps NLQ, 80 col .... call
- Okidata 293 200 cps, 100 cps NLQ, 136 col .... call
- Toshiba P-321 216 cps, 72 LQ .... call
- Toshiba P-351 288 cps, 100 LQ .... call

Letter Quality:
- Epson DX-35 (35 cps) 136 col .... call
- Epson DX-20 (20 cps) 116 col .... save
- Silver-Reed EXP-500 (14 cps) .... call
- NEC-3550 (33 cps) .... call

Math Co-Processors/CPU Chips

- Intel 8087-2 (8 MHz for XT compatibles) $185
- Intel 80287-6 (6 MHz for AT compatibles) $225
- Intel 80287-8 (8 MHz for AT compatibles) $295
- Intel 80287-10 (10 MHz for AT compatibles) $450
- NEC V20-8 (replaces 8088-2) $30

Software

- Microsoft MS-DOS with GW BASIC 3.2 .... $99

In addition to the peripherals listed above we have memory expansion boards, keyboards, floppy drives, RAM Chip Sets, Surge Protectors, Cables, etc.—call for information and prices.

PRICES SUBJECT TO CHANGE WITHOUT NOTICE

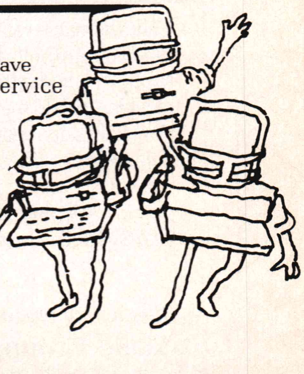
PLEASE CALL FOR LATEST PRICES

Our Famous Guarantee

We were the first mail-order PC outfit to have enough confidence in our products and service to offer this one, and it still can't be beat:

30-Day, Money Back Guarantee; If we can't make our products work for you, we'll gladly take them back (call us for a return authorization). One Year Full Warranty on everything, too. If our products fail, call us for a return authorization and we'll replace them. Free and fast.

It's the best guarantee going, but you probably won't need it.



PC Designs

Circle no. 227 on reader service card.

Call us now: In Tulsa, 1-918-252-5550
In New York, 1-212-514-7280

5837 South Garnett Tulsa, OK 74146 | 19 Rector Street, Suite 2705 New York, NY 10006

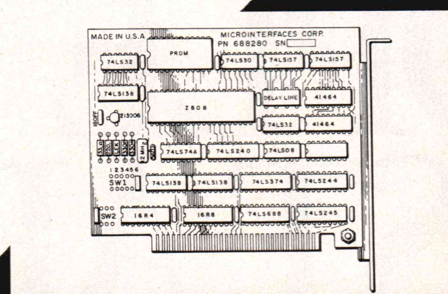
Circle no. 324 on reader service card.

SOLUTIONS!

POWER

The Power to Run Z-80 software on a PC!

Never before has it been easier, or more affordable to add the power of Z-80 compatability. Plug this half size card in your PC, XT, AT or compatible and within minutes you will be running the most powerful CP/M software available including the most popular assemblers, compilers, word processors, spreadsheets, etc. Included with each Z-80 card is our incredible RUN/CPM™ emulation software which will allow your PC to emulate 100's of popular CP/M computers, such as "Kaypro," "Osborne," "Morrow," Etc. Your CP/M programs will run up to 300% faster, with full color, and you can run them from your PC's hard disk or "RAM" disk, and run MS-DOS background programs such as Borland's "Sidekick" on top of your CP/M programs. RUN/CPM™'s amazing disk emulation feature will actually allow your PC's floppy drives to Directly READ, WRITE and FORMAT over 165 different CP/M disks, offering you the utmost in data disk compatability. The RUN/CPM software and Z-80 Co-processor are the perfect solution to bringing the power of the Z-80 to the PC. Call today and order the most affordable Z-80 Co-Processor ever!



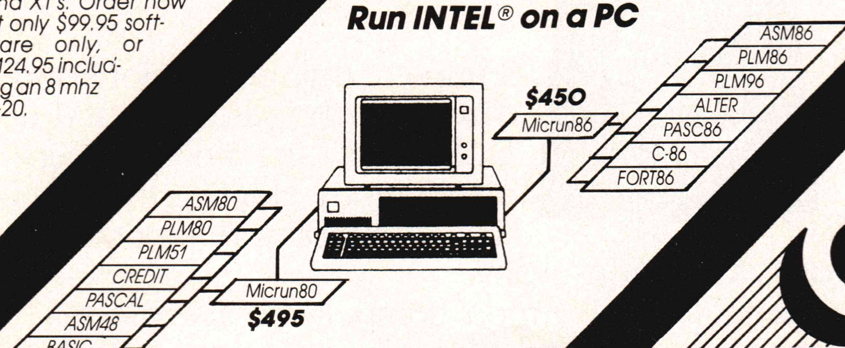
Z-80 Co-processor $199.95

RUN/CPM™ V20/30 Emulator

RUN/CPM™ is also available for the popular N.E.C. V-20 microprocessor chip. This version supports the 8080 instruction set and runs on PC's and XT's. Order now at only $99.95 software only, or $124.95 including an 8 mhz V-20.

COMPATABILITY

Run INTEL® on a PC



Transform an IBM PC, XT, AT, or compatible into an "Intel" Micro Development System, capable of running 8 bit ISIS-II, and 16 bit ISIS-III (UDI) software while remaining 100% MS-DOS compatible. MICRUN 80, our 8 bit ISIS-II Interface is priced at $495 and includes the above pictured Z-80 card plus all necessary software for running 8 bit Intel software on PC's and AT's. MICRUN 86, our 16 bit Interface is $450 and does not require hardware, MICRUN 80 and MICRUN 86 are the solutions to Intel compatability on a PC.

Fast! Hard Disk Back Up.

Quiksave™ is the solution to all of your hard disk back up needs. Quiksave's incredible speed will allow you to back up 10 megabytes of data in only 10 minutes! Because Quiksave backs up data on to standard floppy diskettes, there is no need for an expensive tape drive system. Best of all, Quiksave comes with 20 "Free" pre-formatted floppy disks, so you can start backing up your valuable hard disk data immediately. Don't risk costly data loss, order Quiksave™ today and rest assured.*

**Enough disks to backup 10 Megabytes of data on most systems.*

QUIKSAVE
ONLY **$99.95**

SPEED

Backup 10 Megabytes in 10 Minutes

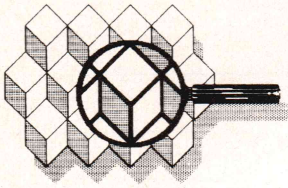
Call to Order, or for more information.
All products come with a 30 Day Money Back Guarantee.

1-800-637-7226

MICRO INTERFACES
6824 N.W. 169th Street
Miami, Florida 33015
(305) 823-8088
Telex 5106004680 MICRO INTER CO.

Circle **no. 110** on reader service card.

OF INTEREST



Languages

MicroMotion has released MasterForth, an implementation of Forth in the Forth-83 standard dialect. It runs on the Macintosh, IBM PC line, Apple II series, Commodore 64, and Z80 CP/M computers, and programs written in MasterForth for one computer can run unchanged on all the others. The product provides a complete programming environment, including a macro assembler and a full-file interface. Relocatable utilities and transient definitions make it possible to run substantial software packages even in a limited memory environment. The string package, screen editor, and resident debugger are standard features. Programs can also be optimized with the optional target compiler. MasterForth is priced at $100—$125. Reader Service No. 16.
MicroMotion
8726 S. Sepulveda Blvd.
#A171
Los Angeles, CA 90045
(213) 821-4340

386/ASM from **Phar Lap Software** is an assembler/linker for the Intel 80386 microprocessor that can be used to create applications for the 80386 on IBM PC, VAX, and Unix host computer systems. A simple extension to the Microsoft .OBJ file format has been created to support the 80386; it allows existing 8086 compilers to be upgraded with minimal effort and to work with the linker supplied with 386/ASM. The 386/ASM package includes the 80386 assembler, the 80386 linker, a users' manual, and examples of 80386 assembly-language programs. It is priced at $495 for the IBM PC version and $4,995 for the VAX/VMS version. Reader Service No. 17.
Phar Lap Software Inc.
60 Aberdeen Ave.
Cambridge, MA 02138
(617) 661-1510

ExperTelligence has announced a new PROLOG for the Macintosh. Exper-Prolog II allows you to load, execute, and modify PROLOG programs interactively. The interpreter includes real numbers, string manipulations, and advanced process control. It's $495. Reader Service No. 18.
ExperTelligence
559 San Ysidro Rd.
Santa Barbara, CA 93108
(805) 969-7874

Smalltalk/V from **Digitalk** includes graphical icons and fonts, a PROLOG compiler, and a source-level debugger. The language is able to perform object swapping to a hard disk or a RAM disk and can handle objects up to 32K in size. It runs on the IBM PC/AT and is priced at $99. Reader Service No. 19.
Digitalk Inc.
5200 W. Century Blvd.
Los Angeles, CA 90045
(213) 645-1082

Simulator-debuggers are available from **Mecklenburg Engineering** that allow you to test and debug object modules for 8-bit microprocessors on an IBM PC. Versions are available for the 63xx, 65xx, 68xx, 8085, 8048, and Z80 processor lines. The simulators read program or data files in hexadecimal format and allow you to run, trace, single-step, and set breakpoints. Status displays show the contents of memory and registers as well as disassembled instructions. It's priced at $75. Reader Service No. 20.
Mecklenburg Engineering
P.O. Box 744
Chagrin Falls, OH 44022
(216) 338-8379

A utility program called EXE2LNK from **Lief Ibsen** allows you to use assembly-language modules within programs written in Logitech's Modula-2. The program converts standard, linked object modules to the .LNK object format. It works with Logitech's Versions 1.10 and 2.0 compilers. It costs DKK 550 ($60). Reader Service No. 21.
Leif Ibsen
Blommevangen 15
DK - 2760 Maalov
Denmark

BES Systems has released a Modula-2 preprocessor that allows programmers to define and use macros as well as include files and conditional compilation. The full source code, written in Logitech's Modula-2/86, is available for $69.95; the compiled utility only is priced at $44.95. Reader Service No. 22.
BES Systems
P.O. Box 270835
Houston, TX 77277
(713) 528-7132

Logical Developments has released RF77, a program that translates Ratfor code into standard FORTRAN. Ratfor is a high-level structured language that eliminates many of the strict requirements of the FORTRAN-77 standard. RF77 creates a standard MS-DOS ASCII file suitable for compilation with most MS-DOS FORTRAN compilers. It's $65. Reader Service No 23.
Logical Developments
P.O. Box 55798
Houston, TX 77255

Tools

PopScreen is a fast screen generator for IBM PCs and compatibles from **Baysoft** that lets you design your displays on screen, with easy access to all the PC's character graphics features. It allows quick creation of boxes, block moves, cursor draws, and individual or global color changes. PopScreen's compacted display data structures can be written as assembly-language source code, in-line code for Turbo Pascal, or as linkable .OBJ code for other linked languages. Displays can also be written to .COM files to be used in batch files or called from DOS. PopScreen costs $39.95. Reader Service No. 24.
BaySoft
P.O. Box 562
Albany, CA 94706
(415) 527-6894

Qlink and Qmake, from **Electrosoft Corp.**, can be used to maintain software for the IBM PC and compatibles. The Qlink linker is fully compatible with Microsoft LINK and operates about ten times faster for the average program. It performs the initial link in a way very much like the way Microsoft LINK and others do, and in addition, it builds a map of the program. Qlink can then build a new executable file by replacing changed object modules and tracking new

THE PROGRAMMER'S SHOP

helps save time, money and cut frustrations. Compare, evaluate, and find products.

RECENT DISCOVERY

Tom Rettig's Library - adds 140 functions to dBASE III Plus for arrays, character and date control, screen, new logical expressions, number manipulations, and much more. Full source (in C, assembler, and dBASE), no royalties. Use with Clipper. PC $ 89

AI-Expert System Dev't

Arity System-incorporate with C programs, rule & inheritance MS $ 279
Experteach - Powerful, no limit on memory size. Samples PC $ 399
EXSYS - Improved. Debug, file & external program access. MS $ 339
Insight 2+ - dB2, language. MS $ 389
LPA MicroProlog Intro w/APES MS $ 149
LPA MicroProlog Prof.w/APES MS $ 595
Others: ESP ($839), Expert Choice ($449)

AI-Lisp

BYSO - Common, MacLISP compatible. 250+ functions, fast PC $ 150
Microsoft MuLisp 85 MS $ 199
PC Scheme LISP - by TI PC $ 95
TLC LISP - classes, **compiler.** MS $ 225
TransLISP - learn fast MS $ 85
Others: IQ LISP ($155), UNX LISP ($59), IQC LISP ($269), WALTZLISP ($149)

AI-Prolog

APT - **A**ctive **P**rolog **T**utor - build applications interactively PC $ 65
ARITY Standard - full, 4 Meg
Interpreter - debug, C, ASM PC $ 350
COMPILER/Interpreter-EXE PC $ 739
With Exp Sys, Screen - KIT PC $1250
LPA MacProlog Complete - incremental compiler and an interpreter MAC $ 295
LPA MicroProlog - intro MS $ 99
LPA MicroProlog Prof. -full memory MS $ 359
Prolog-86 - Learn Fast, Standard, tutorials, samples MS $ 89
Prolog-86 Plus - Develop MS $ 229
TURBO PROLOG by Borland PC $ 69
Others: Prolog-I ($95), Prolog-2 ($839)

AI-Other

METHODS - SMALLTALK has objects, windows, PC $ 69
Personal Consultant by T.I. PC $ 600
Q'NIAL - Combines APL with LISP. Source or binary. PC $ 359
Smalltalk/V-graphics PC $ 89

FEATURES

Baby 34, 36 RPG II by California Software - complete mini RPG environments for PC include compiler, editor (SEU), OCL processor. Screen gen., sort, data exchange, workstation I/O. Separate products for compatibility w/IBM System /38, /36, /34. PC $1250

Unishell - Bourne UNIX shell script compiler translates to C and runs faster. Portable, hand optimizable code. Use as C program generator. PC $ 349

Circle **no. 141** on reader service card.

Compare Products
Use Product Specialists

The Programmer's Shop is much more than the supplier of the largest selection of programmers' software. Trained programming consultants will answer your questions. Ask "product specialists" about Cross Assembler, Translators, Debuggers, or compilers.

Our Services:

- Programmer's Referral List
- Dealers Inquire
- *Compare Products*
- Newsletter
- Help find a Publisher
- Rush Order
- Evaluation Literature FREE
- Over 700 products
- BBS - 7 PM to 7 AM 617-826-4086 National Accounts Center

Basic

Basic Development System - for BASICA; Adds Renum, more. PC $ 105
Basic Development Tools by Sterling Castle PC $ 99
Basic Windows by Syscom PC $ 95
BetterBASIC - all RAM, modules
Structure. Full BASICA PC $ 165
8087 Math Support PC $ 89
Run-time Module PC $ 229
Better Tools - for Better Basic PC $ 95
CADSAM FILE SYSTEM - full ISAM in MBASIC source. MS $ 75
GoodBas - maintain code PC $ 95
LPI Basic - MS compatible UNIX $1100
Prof. Basic - Interactive, debug PC $ 79
8087 Math Support PC $ 47
QuickBASIC V2.0-New interface PC $ 67
TRUE Basic - ANSI PC $ 109
Run-time Module PC $ 459

Cobol

LPI Cobol - ANSI '74 UNIX $1200
Macintosh COBOL - full MAC $ 459
MBP - Lev. II, native MS $ 885
Microsoft Cobol Tools - xref, debugger w/source support. Xenix $359 PC $ 219
Microsoft Version II - upgraded. Full Lev. II, native, screens. MS $ 479
Professional COBOL PC $2695
Realia - very fast MS $ 839
Ryan McFarland COBOL MS $ 699
COBOL-8X MS $ 995
VS Workbench PC $3500

Editors for Programming

BRIEF Programmer's Editor - undo, windows, reconfigure PC Call
EMACS by UniPress - powerful, multifile, windows Source:$929 $ 299
Epsilon - like EMACS, full C-like language for macros. PC $ 169
KEDIT - like XEDIT PC $ 109
Lattice Screen Editor - multiwindow, multitasking Amiga $100 MS $ 109
PC/VI - Custom Software PC $ 129
PMATE - power, multitask PC $ 149
SPF/PC - fast, virtual memory PC $ 139
XTC - multitasking PC $ 85

C Libraries-Communications

Asynch by Blaise PC $ 149
Greenleaf Comm Lib. PC $ 149
Multi-Comm - add multitasking, use w/Multi-C PC $ 149
Software Horizons pack 3 PC $ 119

RECENT DISCOVERY

Pdisk - Improve your PC's performance with disk cacheing, extend DOS with single command tree structure manipulation and file MOVE, and maintain your hard disk with Advanced Backup/Restore. PC $159

C Language-Compilers

AZTEC C86 - Commercial PC $499
C86 by CI - 8087, reliable MS $299
Datalight C - fast compile, good code, 4 models, Lattice compatible, Lib source. Dev'rs Kit PC $ 79
HOT C - new, intriguing PC $ 85
Lattice C - from Lattice MS $299
Mark Williams - w/debugger MS $399
Microsoft C 4.0- Codeview MS $279
Wizard C - Lattice C compatible, full sys. III, lint, fast. MS $389

C Language-Interpreters

C-terp by Gimpel - full K & R MS $239
C Trainer by Catalytix PC $ 89
INSTANT C - Source debug, Edit to Run-3 seconds, .OBJs MS $389
Interactive C by IMPACC Assoc. Interpreter, editor, source, debug. PC $225
Introducing C-self paced tutorial PC $109
Run/C Professional MS $189
Run/C Lite - improved MS $109

C Libraries-General

Blackstar C Function Library PC $ 79
C Essentials - 200 functions PC $ 85
C Food by Lattice-ask for source MS $109
C Scientific Subroutines - Peerless MS $139
C Tools Plus (1 & 2) PC $139
C Utilities by Essential - Comprehensive screen graphics, strings, source. PC $139
C Worthy Library - Complete, machine independent MS $269
Entelekon C Function Library PC $119
Entelekon Superfonts for C PC $ 45
Greenleaf Functions-portable, ASM $139
PforCe by Phoenix - objects PC $299

C Libraries-Files

FILES: C Index by Trio - full B+ Tree, vary length field, multi compiler
/File is object only MS $ 89
/Plus is full source MS $349
CBTREE - Source, no royalties MS $ 99
CTree by Faircom - no royalties MS $339
dbQUERY - ad Loc, SQL - based MS $159
dbVISTA - full indexing, plus optional record types, pointers, Network.
Object only - MS C, LAT, C86 $159
Source - Single user MS $429
Source - **Multiuser** MS $849
dBASE Tools for C PC $ 79
dbc Isam by Lattice MS $199
dBx - translator MS $319
w/source MS $519

Atari ST & Amiga

We carry full lines of Manx, Lattice, & Metacomco.
Amiga - LINT by Gimpel Amiga $ 79
Cambridge LISP Amiga $200
Lattice C ST, Amiga $139
Lattice Text Utilities Amiga $ 75
Megamax - tight, full ST $200

We support MSDOS (not just compatibles), PCDOS, Xenix-86, CPM-80, Macintosh, Atari ST, and Amiga.

THE PROGRAMMER'S SHOP

provides complete information, advice, guarantees and every product for Microcomputer Programming.

"COBOL FOR PC-DOS" SAVINGS

Order before 12/31/86 and mention this ad for the special prices to the right.

GSA High Level ANSI '74 standard plus compatibility with respected complete COBOL environment.

	List	Before 12/31	After 12/31
Microfocus Level II High Performance COBOL	$1500	$ 825	$1470

OR: Buy one of these Microfocus compilers bundled with Sourcewriter and receive extra savings off our regular discounted price.

with Sourcewriter	List	Normal Discount	Special Savings
Level II High Performance	$3500	$3319	$1789
Professional COBOL	$5000	$4544	$3492
VS Workbench	$6000	$5349	$4289

C Support-Systems

Basic-C Library by C Source PC $139
C Sharp - well supported. Source, **realtime,** tasks, state system PC $600
C ToolSet - DIFF, xref, source MS $ 95
The HAMMER by OES Systems PC $179
Lattice Text Utilities MS $ 95
Multi-C - multitasking PC $149
PC LINT-Checker. Amiga $89 MS $119
SECURITY LIB-add encrypt to MS C, C86 programs. Source $250 PC $125
Unishell - script compiler PC $349

C-Screens. Windows. Graphics

C Power Windows by Entelekon PC $119
dBASE Graphics for C PC $ 79
Curses by Lattice PC $ 99
ESSENTIAL GRAPHICS - fast, fonts, no royalties PC $209
GraphiC - mono version PC $219
GraphiC - new color version PC $299
Multi-Windows - use w/ Multi-C PC $295
Topview Toolbasket by Lattice PC $209
View Manager for C by Blaise PC $219
Vitamin C - screen I/O PC $139
Windows for C - fast PC $159
Windows for Data - validation PC $239
ZView - screen generator MS $199

Debuggers

Advanced Trace-86 by Morgan Modify ASM code on fly. PC $139
CODESMITH - visual, modify and rewrite Assembler PC $109
C SPRITE - data structures PC $139
DSD87 - by Soft Advances PC $ 89
Periscope I - own 16K PC $249
Periscope II - symbolic, "Reset Box," 2 Screen PC $119
Pfix-86 Plus Symbolic Debugger by Phoenix - windows PC $249
Showcase by Test Software PC $135
Software Source by Atron - Lattice, MS C, Pascal, Windows single step, 2 screen, log file. MS $115
w/Breakswitch $199

RECENT DISCOVERY

MKS Toolkit - Unix-type tools for PCDOS include popular **vi,** 'awk' report generation language, profiler, grep, diff, much more; over 70 utilities in all. PC $119

Fortran & Supporting

ACS Time Series MS $429
Forlib + by Alpha - graphics and file routines, comm. MS $ 59
MACFortran by Microsoft MAC $229
MS Fortran link to C MS **$209**
No Limit - Fortran Scientific PC $119
RM/Fortran - enhanced "IBM Professional Fortran" MS $395
Scientific Subroutines - Matrix MS $149
Statistician by Alpha MS $259
Strings and Things - register, shell PC $ 59

Multilanguage Support

BTRIEVE ISAM MS $199
BTRIEVE/N-multiuser MS $469
CODESIFTER - Profiler. MS $ 99
HALO Graphics - 115 + devices. Animation, engineering, business. Any MS language, Lattice, C86 PC $219
Informix - by RDS PC $689
Informix 4GL - application builder PC $849
PANEL - Create screen with editor, generates code. Full data validation, no royalties. Xenix $539, MS $229
PolyLibrarian by Polytron MS $ 85
PVCS Version Control MS $329
QMake by Quilt Co. MS $ 89
Rtrieve - Xtrieve option MS $119
Screen Sculptor - slick, thorough, fast, BASIC, PASCAL. PC $ 99
Xtrieve - organize database MS $199
ZAP Communications - VT 100, TEK 4010 emulation, file xfer. PC $ 95

Pascal and Supporting

ALICE - learn Pascal, Turbo compatible, interpreter PC $ 79
Exec - Chain Programs MS $ 85
MetaWINDOW - graphics toolkit PC $139
Microsoft PASCAL - faster MS $199
MICROTEC PASCAL - 5 memory models, "Iterators", 65 bit 8087 strings MS $665
Pascal Pac with Tidy - formatter, utilities PC $ 69
Pascal Tools - strings, screen PC $109
Pascal Tools 2 - by Blaise MS $ 85
Pfas - Portable Isam MS $185
TurboHALO - 150 routines, IBM EGA, Hercules, more PC $ 85
USCD Pascal - native code MS $ 69

Circle **no. 133** on reader service card.

RECENT DISCOVERY

Pascal-2 - Perhaps tightest compiler for MSDOS. Mainframe background & power. MS compatible. Complete environment. Turbo translator. Get detailed specs. PC $ 329

Other Languages

APL*PLUS/PC PC $ 469
Artek ADA Compiler - DOD standard minus multitasking PC $ 469
CLIPPER-dBASE Compiler MS $ 449
Correct FORTH - ROMable PC $ 80
ED/ASM - 86 by Oliver PC $ 85
Lattice RPG II Compiler PC $ 639
MacASM - fast MAC $ 99
MasterForth - Forth '83 MAC or PC $ 125
Microsoft MASM - faster MS $ 98
Modula 2 by Volition Systems MS $ 250
Modula-2/86 Compiler by Logitech w/ 8087 ($105), 512K ($149). PC $ 65
Pasm - by Phoenix MS $ 149
SNOBOL4 + - great for strings MS $ 85
Turbo Edit/ASM - by Speedware PC $ 85

Xenix-86 & Supporting

Basic - by Microsoft $ 279
Cobol - by Microsoft $ 795
Fortran - by Microsoft $ 589
MicroFocus Lev. II Compact COBOL $ 899
Xenix Complete Development System $1149

Other Products

386 Assembler/Linker PC $ 459
ASMLIB - 170 + routines PC $ 135
BSW Make - like UNIX make MS $ 85
Dan Bricklin's Demo Program PC $ 65
dBrief - Customize BRIEF for dBASE development. with BRIEF $275. PC $ 95
H Test/H Format - XT Fix PC $ 89
Interactive Easyflow-HavenTree PC $ 129
Link & Locate - tools to work with Intel and Tektronix projects. MS $ 329
LMK - like UNIX make MS $ 149
Microsoft Windows PC $ 75
Microsoft Windows Software Development Kit PC $ 399
Opt Tech Sort - sort, merge MS $ 119
PMaker - by Phoenix PC $ 99
Polymake by Polytron MS $ 79
PS MAKE by UniPress MS $ 79
SECRET DISK by Lattice PC $ 95
SET:SCIL - manager revisions PC $ 299
Shrink/Shrinkem - put more files on disk with spacemaker PC $ 150
SoftEst - Manage projects. MS $ 350
Synergy-Create user interfaces MS $ 375
Texsys - control source MS $ 89
Visible Computer: 8088 - Simulates demos or any .exe. com, Debugger. 350 pg. tutorial, unprotected PC $ 69

Note: All prices subject to change without notice. Mention this ad. Some prices are specials. Ask about COD and POs. Formats: 3" laptop now available, plus 200 others. UPS surface shipping add $3/item.

Call for a catalog, literature, advice and service you can trust

HOURS
8:30 AM - 8:00 PM EST.

800-421-8006

THE PROGRAMMER'S SHOP™
128-L Rockland Street, Hanover, MA 02339
Mass: 800-442-8070 or 617-826-7531 10/86

"I have been pleased with your catalog selection, the knowledge of your telephone answering staff, and the promptness of your service. I have every wish to become a continuing customer of The Programmer's Shop."

Carl C. Rollo

OF INTEREST
(continued from page 130)

ones instead of relinking the entire program. Qmake is similar to the Unix make utility. It follows a map of the program and looks at the file generation times to tell if an output file needs to be constructed based on the component source files. Qmake allows a single executed command to process all the files that need to be updated. The programs are not copy-protected. Qlink costs $250, Qmake costs $100, and a package containing both costs $295. Reader Service No. 25.
Electrosoft Corp.
3003 Washtennaw Ave.
Ste. 1
Ann Arbor, MI 48104
(313) 973-1229

TDebug PLUS, from **TurboPower Software**, allows Turbo Pascal programmers to trace, set breakpoints, and examine and set variables at run time using the symbolic variables from Pascal source code. TDebug PLUS automatically loads Turbo Pascal, which operates normally for editing and compiling the program. After the program has compiled, TDebug PLUS takes control. The source code is displayed on one half of the screen, and the other half displays the debugging commands and responses. TDebug PLUS's MAP files are fully compatible with many external assembly-language debuggers. TDebug runs with Turbo Pascal 3.0 and costs $60. Reader Service No. 26.
TurboPower Software
3109 Scotts Valley Dr.
Ste. 122
Scotts Valley, CA 94066
(408) 438-8608

Artificial Intelligence

A fully functional LISP library and programming environment for the C language is now available for the IBM PC line and either Microsoft or Lattice compilers from **Frederick J. Drasch Computer Software.** Written entirely in C, Clisp is Common LISP-compatible and consists of more than 100 functions, including all LISP primitives, predicates and conditionals, association and property lists, symbolic pattern matcher, context-sensitive database and stack, and an interpreter with hooks to add user-written functions. It sells for $189 (with source code). Reader Service No. 27.
Frederick J. Drasch Computer Software
RFD#1 Box 202
Ashford, CT 06278
(203) 429-3817

Gold Hill Computers has introduced the GCLISP 386 Developer and the 386 HummingBoard for IBM PCs and compatibles. The GCLISP 386 Developer is a powerful Common LISP development environment for 386-based systems. It addresses up to 15 megabytes of memory and includes a large memory interpreter and compiler, editor, tutorial, and on-line help system and supports lexical scoping, packages, and transcendental functions. The 386 HummingBoard is an Intel 386-based plug-in board with memory that is specifically tailored to run large LISP applications quickly. Directly addressable on-board memory is expandable to

(continued from page 134)

24 megabytes with 1-megabit DRAMs (up to 6 megabytes with 256K DRAMs). GCLISP 386 Developer costs $1,195; when bundled with the HummingBoard, it costs $7,000. Reader Service No. 28.
Gold Hill Computers
163 Harvard St.
Cambridge, MA 02139
(617) 492-2071

TEX for Micros

Addison-Wesley has released a complete IBM PC implementation of Donald Knuth's TEX typesetting system. MicroTEX is a "mark-up" system, meaning that users add special instructions to a standard ASCII file. The program then creates its own file that can be used to preview the typeset text on the screen. It costs $295 alone; with an Epson/IBM print driver, it costs $369; and with a PostScript driver, it costs $495. Reader Service No. 29.
Addison-Wesley
Publishing Co.
Jacob Wy.
Reading, MA 01867
(617) 944-3700

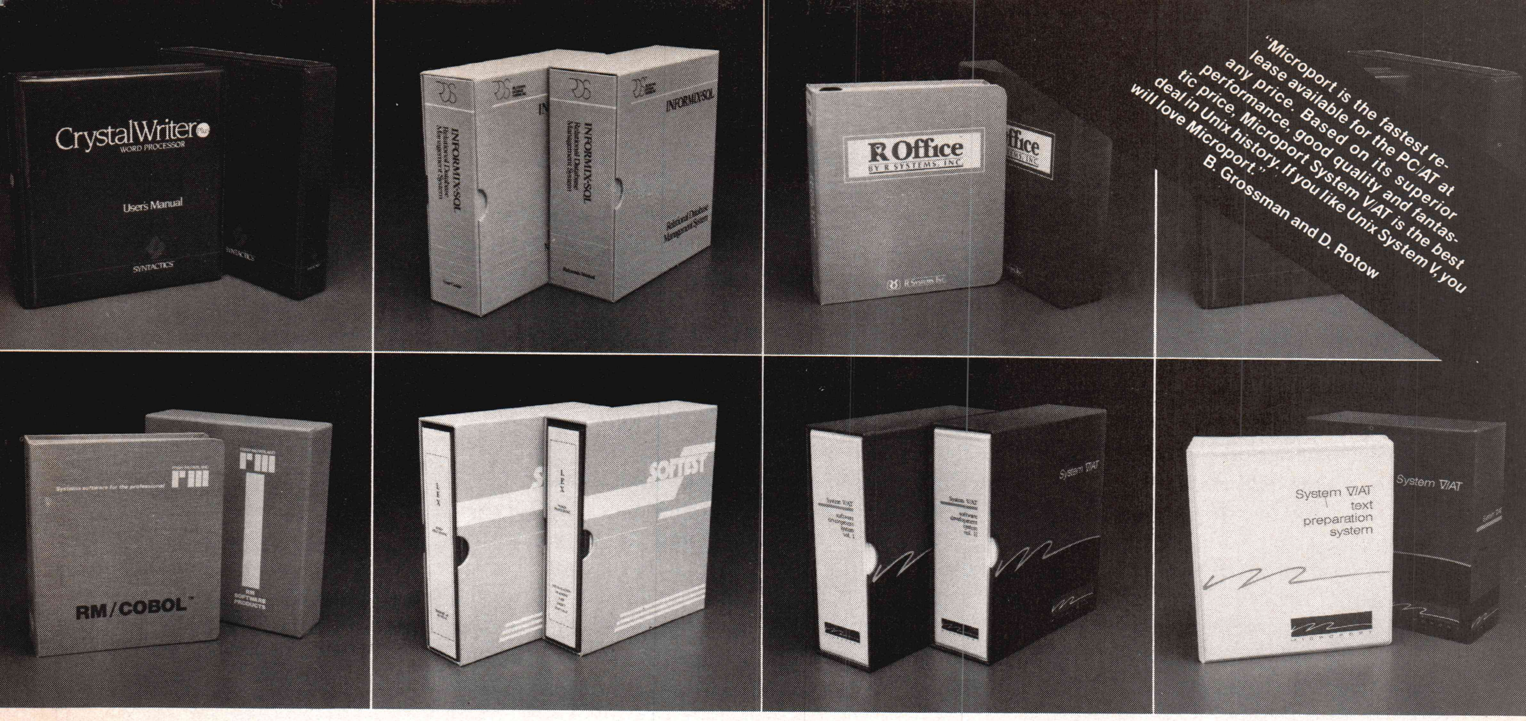
FTL Systems has unveiled MacTeX, a desktop typesetting software package for the Macintosh. MacTeX combines TEX with the graphics power of PostScript. It offers a full set of professional typesetting features and creates camera-ready typeset documents on most PostScript-compatible typesetters and printers. MacTeX also provides a built-in, multiwindow text editor with MacWrite-type functions. MacPaint images can be imported and scaled, and 248 PostScript commands are supported. A preview mode allows for on-screen WYSIWYG viewing of the document text. MacTeX runs on the Mac Plus and costs $750. Reader Service No. 30.
FTL Systems Inc.
234 Eglinton Ave. E, Ste. 205
Toronto, Ont.
Canada M4P 1K5
(416) 487-2142

For the Macintosh

Peachtree Technology has released its first product for the Macintosh, a 20-megabyte hard-disk drive called the S-20 Plus that is two to six times faster than the Apple HD-20 drive. The external 3.5-inch drive uses the Mac Plus SCSI port. It costs $1,395. Reader Service No. 31.
Peachtree Technology
3120 Crossing Pk.
Norcross, GA 30071
(404) 662-5158

The TDBK-20+ from **MDIdeas** is a tape backup unit that lets you back up your Mac Plus hard disk quickly. The TDBK-20+ utilizes the SCSI port and can back up 22 megabytes on each tape. It can back up and restore virtually any Macintosh hard disk or volume. All data is read and verified to ensure reliability and integrity of all backups. Even copy-protected software, which normally can be installed on a hard disk only once, can be properly restored. The TDBK-20+ requires a Macintosh equipped with a SCSI port and costs $1,095. Reader Service No. 32.
MDIdeas Inc.
1111 Triton Dr.
Ste. 205
Foster City, CA 94404

WHY BUY OUR $159 UNIX SYSTEM FOR YOUR PC/AT WHEN YOU CAN GET THE SAME THING FOR A THOUSAND DOLLARS MORE?



HERE ARE THE REASONS WHY.

Best-price UNIX™ applications software. From the company that made a reality of best-price UNIX operating systems.

Business computing, word processing, database management, text preparation and software development—we've selected, developed and added value to the best. Once you've converted your PC/AT, with our System V, into a true multi-user, multi-tasking machine, you'll demand software that lets you get on with the tasks at hand.

CRYSTALWRITER PLUS

is a comprehensive word-processing system for the multi-user office. It allows concurrent use of files, layouts and printers, as well as supporting a full range of terminals.

INFORMIX SQL

is the relational-database-management system developers will choose for custom applications. It offers relational-database flexibility, interactive-database queries, a flexible design environment, automated functional menus, custom screen design and total format control.

UNIX and DWB are trademarks of AT&T
IBM and IBM PC-AT are trademarks of IBM CORPORATION.
SYSTEM V/AT is a trademark of MICROPORT SYSTEMS, INC.

R-OFFICE

ties a powerful word processor together with the file-management, table-spreadsheet, report-generation and desktop-management chores common to nearly every modern office.

R/M COBOL

is Ryan-McFarland's GSA certified implementation of the ANSI X3.23 74 standard. R/M COBOL is designed for serious business computing.

LEX

is a standalone system offering complete word processing capabilities, as well as utilities for adapting LEX to any type of video terminal or printer.

SOFTWARE DEVELOPMENT

from Microport. Our package was derived from the extensively-tested Portable C compiler. (The large-model code produced by the compiler is among the densest and fastest available.) These are true 286 compilers, making full and efficient use of the extended 286 instruction set. The package also includes the F77 Fortran compiler.

TEXT PREPARATION

from Microport. This package consists of the complete System V, Release 2 Documenter's Workbench (DWB), and includes both the new troff and the old troff. Drivers for the HP Laser Jet Printer and Apple's LaserWriter are also available.

These are but a few of our available packages. Virtually everything listed in AT&T's catalog can now, or will soon run with our operating system.

Regarding Our SYSTEM V operating system.

More than a look-alike, Microport's System V complies with AT&T's new binary standard and was derived from AT&T's *own* UNIX System V, Release 2 iAPX286. It thereby contains standard System V features the competitors don't support, such as the powerful symbolic debugger (SDB), the shell-layering job-control facility and the F77 Fortran compiler, as well as programming tools such as ctrace, cflow and bs. Over 200 utilities come standard to let you search and modify files, make use of electronic mail, calculate electronically, convert data and publish.

- **90 day money-back guarantee**
- **Educational site licensing**
- **Commitment to AT&T source and binary standards**

To order: Call any of our numbers below for today's price and product line quote (be sure to ask about our buy-two-applications-packages-and-get-RUNTIME-free offer).

You really wouldn't want to pay thousands of dollars more, would you?

M I C R O P O R T

MICROPORT SYSTEMS, INC.
4200 Scotts Valley Drive
Scotts Valley, CA 95066
408/438-UNIX or 408/438-8649 (CA)
or 800/PC2-UNIX
TELEX 268/863-3650

Circle no. **154** on reader service card.

(continued from page 136)

(415) 573-0580

For the Amiga

ENVISAGE is a multimedia periodical for the Amiga computer from **Chestnut Computer Graphics and Sound.** Four times a year, subscribers receive a magazine, a stereo cassette tape, color photos of Amiga graphics, an Amiga disk, and a collection of product information flyers and advertisements for Amiga related products. They also receive access to a computer bulletin-board system. An annual subscription to ENVISAGE costs $60. Reader Service No. 33.

Chestnut Computer
Graphics and Sound
P.O. Box 417
Hatfield, PA 19440
(215) 855-0741

A 512K memory-expansion box called Alegra is available for the Amiga from **Access Associates**. The unit plugs into the expansion slot and consumes less that 5 watts. It allows for future upgrades to 2 megabytes and supports the Amiga's auto-configuration architecture. The price is $379 for the 512K version. Reader Service No. 34.

Access Associates
491 Aldo Ave.
Santa Clara, CA 95054
(408) 727-0256

For the Atari ST

A multitasking operating system for the ST is available from **Beckmeyer Development Tools.** Called Micro RTX, it is fully compatible with TOS and even runs unmodified TOS programs. The number of active tasks is limited only by the amount of available memory. Micro RTX is priced at $69.95. Reader Service No. 35.

Beckmeyer Development
Tools
592 Jean St., #304
Oakland, CA 94610
(415) 658-5318

Mach2 from **Palo Alto Shipping,** is a multitasking Forth-83 development system for the Atari ST. Like other implementations of Forth, Mach2 is highly interactive, allowing you to experiment with the ST without going through the compile-link-execute cycle. The Mach2 package includes an integrated GEM editor, full GEM and TOS support, Motorola assembler, examples, and a 300-page manual. It costs $59.95. Reader Service No. 36.

Palo Alto Shipping
P.O. Box 7430
Menlo Park, CA 94062
(415) 854-2749
(800) 44-FORTH

Miscellaneous

SideTalk from **Lattice** is a package of telecommunications programs for the IBM PC plus a communications programming language. The package can be used as a memory-resident program or alone. The price is $119.95. Reader Service No. 37.

Lattice Inc.
P.O. Box 3072
Glen Ellyn, IL 60138
(312) 858-7950

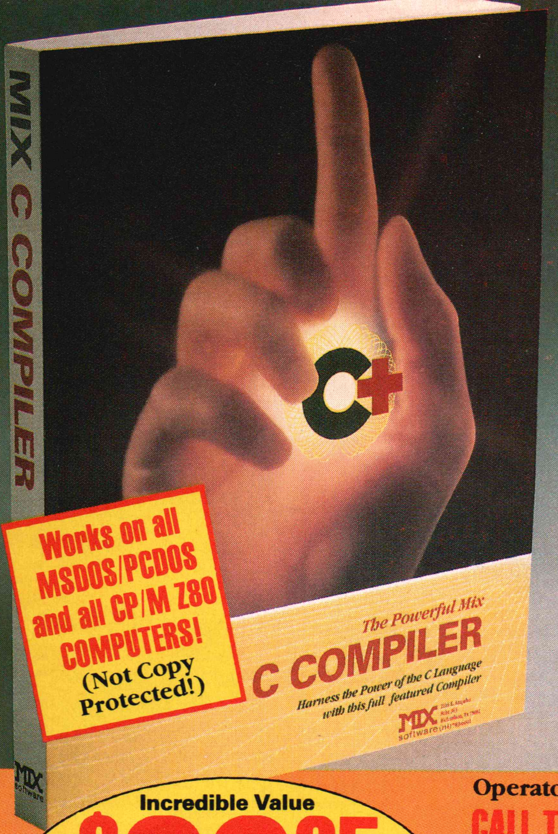
A full version of Unix V is available for the IBM PC/AT from **Microport Systems.** System V/AT contains more than 200 utility programs. The implementation includes a virtual-screen windowing system, FORTRAN and C compilers, and a symbolic debugger. The file system can be "hardened" against power outages through special disk software. System V/AT costs $159 for a run-time package; a software development system with full language support costs $169; and a site license for educational institutions is available for $800. Reader Service No. 38.

Microport Systems Inc.
10096 Soquel Dr.
Aptos, CA 95003
(408) 688-0286

A multitasking kernel for Z80, NSC800, and HD64180 microprocessors is available from **Echelon.** Called QUICK-TASK, it includes full source code. The kernel is a ROMable code segment of less than 4K that supports up to 255 tasks with hardware interrupt support and multiprocess synchronization. The full package costs $249. Reader Service No. 39.

Echelon Inc.
885 N. San Antonio Rd.
Los Altos, CA 94022
(415) 948-3820

DDJ



C FEVER *catch it!*

It's becoming an epidemic . . . everyone is switching to C! First there were a few hackers, then came the college students, next the major software houses, and now the rest of the programming world. Programmers everywhere are infected with the desire for SPEED, POWER, and PORTABILITY.

It's time to face the inevitable. You're going to catch the fever too! When you do, give us a call. We've got the best cure—an illustrated guide to the C language, plus a complete program development system. Everything you need to master the C programming language . . . all at a price that's less than the cost of a book!

But don't let this price fool you. Our system is powerful; it compiles twice as fast as the others, is completely standard, and it's very easy to use. Most C compilers are designed for wizards. We have designed ours for you!

What do you get for a mere $39.95?

- **A 450 Page book filled with sample programs, plus . . .**
- **A fast, standard, full featured C compiler** that supports all data types and the latest features like bit fields, enumerations, structure assignment, and passing/returning structures.
- **A fast linker** that loads separately compiled files, searches libraries, and builds an executable program.
- **An extensive library of more than 170 functions** (including the standard C functions and the computer specific functions that provide direct access to the operating system and BIOS).
- **Tools that allow you to optimize** your programs for minimal space or maximum speed.

Incredible Value
AT ONLY $39.95 With 30 Day Money-Back Guarantee

Operators are standing by . . . Please use this Number for ORDERS ONLY!

CALL TOLL FREE FOR RUSH ORDER DELIVERY!

1-800-523-9520

IN TEXAS, PLEASE CALL TOLL FREE 1-800-622-4070

For Technical Support Please call 1-214-783-6001

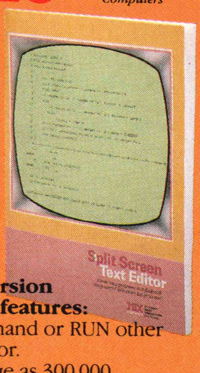
MIX Software, Inc. / 2116 E. Arapaho / Suite 363 / Richardson, Texas 75081

Or contact our Worldwide Distributors direct in:

Canada:	Saraguay Software	1-416-923-1500	Switzerland:	DMB Communication	CH-1-825-53-29
Australia:	Techflow	047-586924	France:	Info/Tech	1-43-44-06-48

Split Screen Text Editor

an Incredible Value AT ONLY $29.95 *(Not Copy Protected!) Works on all MSDOS/PCDOS and CP/M Z80 Computers*



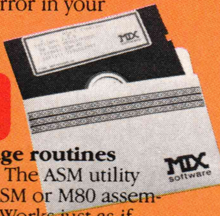
Our high powered editor is great for editing high level languages. **It works just like Micropro's Wordstar,™** but macros allow you to create your own custom editor, and the split-screen feature lets you edit two files at the same time.

The MSDOS/PCDOS version is loaded with special features:

- Execute any DOS command or RUN other programs from the editor.
- Quickly edit files as large as 300,000 characters.
- Compile MIX C programs directly from memory. The editor automatically positions the cursor to the first error in your program.

ASM UTILITY

an Incredible Value AT ONLY $10



Call assembly language routines from your C programs. The ASM utility works with Microsoft's MASM or M80 assembler. Macros make it easy! Works just as if you were calling a C function, and you can even call C functions from assembly language. Lots of useful assembly language functions are included as examples.

RUSH REPLY ORDER FORM!

Description	Quantity	PRICE	Total Order
Split-Screen Text Editor	______	$29.95	$______
C Compiler	______	$39.95	$______
C and Text Editor (Special)	______	$54.95	$______
ASM Utility	______	$10.00	$______
Texas Residents Add 6.125% Sales TAX			$
Shipping Charges (See at Right)			$
TOTAL OF YOUR ORDER:			$

Please check method of payment:
☐ Check ☐ Money Order ☐ MasterCard/VISA

Your Card #: ______

Expires ______

Shipping Charges: (No charge for ASM Utility)
In the U.S.A.: Add $5.00 per Order.
In CANADA: Add $10.00 per Order.
OVERSEAS: Add $10.00 per Text Editor. Add $20.00 per C Compiler. Add $30.00 for combined C Compiler and Text Editor.

Operating System: ***(Check one)***
☐ CP/M Z80 ☐ MSDOS/PCDOS

Specify Your Computer Name ______

Specify Disk Format ______

NAME ______

Telephone A/C (______) ______

Street ______

City ______

State ______

Country ______ ZIP ______

MIX software
2116 East Arapaho
Suite 363
Richardson, Texas, 75081

Ask about our Volume Discounts!
Call 1-214-783-6001

D

Circle no. **156** on reader service card.

LOGITECH MODULA-2/86 HOLIDAY PACKAGE

$89 Price

- Separate Compilation
- Native Code Generation
- Large Memory Model Support
- Multitasking
- Powerful Debugging Tools
- Comprehensive Module Library
- Available for the PC and the VAX

Use LOGITECH MODULA-2/86 to decrease your overall development cycle and produce more reliable, more maintainable code.

LOGITECH MODULA-2/86 — $89

Includes Editor, Run Time System, Linker, 8087 Software Emulation, Binary Coded Decimal (BCD) Module, Logitech's comprehensive library, Utility to generate standard .EXE files. AND more!

LOGITECH MODULA-2/86 with 8087 Support — $129

LOGITECH MODULA-2/86 PLUS — $189

For machines with 512K of RAM. Increases compilation speed by 50%.

RUN TIME DEBUGGER (Source level!) — $69

The ultimate professional's tool! Display source, data, call chain and raw memory. Set break points, variables, pinpoint bugs in your source!

UTILITIES PACKAGE — $49

Features a Post-Mortem Debugger (PMD). If your program crashes at run-time the PMD allows you to analyze the status of the program and locate the error. Also includes a Disassembler, Cross Reference Utility, and Version that allows conditional compilation.

LIBRARY SOURCES — $99

Source code now available for customization or exemplification.

WINDOW PACKAGE — $49

Build windows into your programs. Features virtual screens, color support, overlapping windows and a variety of borders.

MAKE UTILITY — $29

Figures out dependencies and automatically selects modules affected by code changes to minimize recompilation and relinking.

CROSS RUN TIME Debugger and ROM Package — $199

Still available at an introductory price!

TURBO PASCAL to MODULA-2 TRANSLATOR — $49

"Turbo Pascal . . . is a very good system. But don't make the mistake of trying to use it for large programs."

*Niklaus Wirth**

Our Translator makes it even easier for Turbo users to step up to Modula-2/86. It changes your Turbo source code into Modula-2/86 source, solves all the incompatibilities, and translates the function calls of Turbo into Modula-2/86 procedures. Implements the complete Turbo libraries!

Call for information about our VAX/VMS version, Site License, University Discounts, Dealer & Distributor pricing.

30 Day Money Back Guarantee!

To place an order call our special toll free number: **800-231-7717**

in California **800-552-8885**

$199 Special Holiday Offer

Step up to the power of LOGITECH MODULA-2/86 at a saving of nearly $100 off our usual low prices! We're offering a complete tool set including our compiler with 8087 support (for use with or without an 8087), our Turbo to Modula-2/86 Translator, Run Time Debugger, and Utilities in one holiday package at a special price!

YES I want to step up to LOGITECH MODULA-2/86!

Here's the configuration I'd like:

☐ Special Holiday Package	$199
☐ Logitech Modula-2/86	$89
☐ with 8087 support	$129
☐ Plus Package	$189
☐ Turbo to Modula Translator	$49
☐ Run Time Debugger	$69
☐ Utilities Package	$49
☐ Library Sources	$99
☐ Window Package	$49
☐ Make Utility	$29
☐ ROM Package	$199

Add $6.50 for shipping and handling. Calif. residents add applicable sales tax. Prices valid in U.S. only.

Total Enclosed $______

☐ Visa ☐ MasterCard ☐ Check Enclosed

Card Number ______ Expiration Date ______

Signature ______

Name ______

Address ______

City ______ State ______ Zip ______

Phone ______

LOGITECH

Logitech, Inc.
805 Veterans Blvd.
Redwood City, CA 94063
Tel: 415-365-9852

In Europe:
Logitech SA, Switzerland
Tel: 41-21-879656

In Italy: Tel: 39-2-215-5622

*as reported in Micro Cornucopia, August-September 1985. Turbo Pascal is a registered trademark of Borland International.

ADVERTISER INDEX

*This advertiser prefers to be contacted directly: see ad for phone number.

ADVERTISING SALES OFFICES

Midwest
Michele Beaty **(317) 875-8093**

Southeast
Gary George **(404) 897-1923**

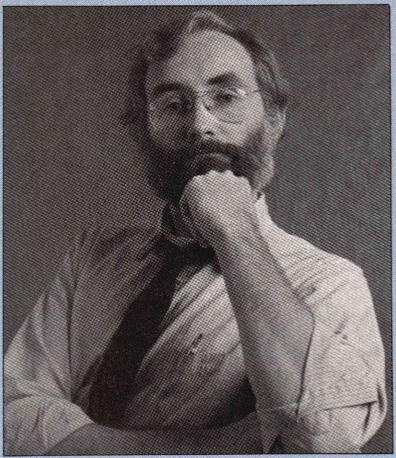
Northeast
Cynthia Zuck **(718) 499-9333**

Northern California/Northwest
Lisa Boudreau **(415) 366-3600**

Southern California/AZ/NM/TX
Michael Wiener **(415) 366-3600**

Advertising Director
Robert Horton **(415) 366-3600**

SWAINE'S FLAMES



Ray Duncan, Allen Holub, and I flew up to Redmond, Washington in October for a Microsoft seminar on languages. Navigating the halls, we had to dodge the pushcarts loaded with newly arrived 80386 machines. There was an AT with EGA on every desk. A four-building network that works. Sixty Sun workstations. Yeah, but they get a lot of rain.

One feature of the seminar was a "Storm the Gates" programming contest. Frivolous though it was, I must admit that it was also a good show and that Bill Gates finished well ahead of all challengers.

Microsoft is concerned just now with competition of a more commercial sort from Borland International. QuickBASIC, Version 2, recently became Microsoft's fastest-selling product. Microsoft is pushing QB2, its shot at Borland's market, like no other product in the company's history.

What is Borland's market, as Microsoft sees it? Microsoft language marketing manager Rob Dickerson polled QuickBASIC 1.0 and Turbo Pascal users. In each case, most used the product at work, and they usually employed it for data management. A picture of a business user writing small report programs suggests itself.

Well, maybe. Meanwhile back in Scotts Valley, Borland has listened to its market and released a new version of Turbo Prolog. Is this the same Borland market? Perhaps a BASIC and a Pascal can share one market, but who are the users of Turbo Prolog, and does any of this have anything to say about who will buy a Turbo C or QuickC?

I ask that question because Microsoft and Borland are at work on Turbolike C compilers. Bill Gates said at the conference that he wants to see Borland bring out a Turbo C, and I'm sure that Phillipe Kahn will be sent one of the first beta copies of QuickC. One market for QuickC could be Windows developers looking for a fast prototyping tool. I'm sure that either company could do well with a Turbo-like C compiler, but I can't believe that the audience is Turbo Pascal programmers.

(Another product that may make Windows development more appealing is a third-party 34010 graphics board that will be announced shortly after my deadline for this column.)

One of the most demanding markets for Microsoft's C compilers is Microsoft, and one of the most important projects underway at Microsoft just now is the work on optimization. Since Microsoft does all its programming in C and assembly language now, optimizing its own C compilers should result in immediate improvements across its entire product line.

What makes serious optimization promising is the 80386 chip, as global dataflow analysis has until now proved too demanding for microprocessor power. What's scary for Microsoft is a new kind of competition. Rob Dickerson admits to being worried about the minicomputer compiler vendors who already have powerful optimizing compilers. Some of these will surely survive the radical paradigm shift into the microcomputer market and make life interesting for microcomputer compiler vendors.

It's not clear whether Microsoft is worried by operating system competitors in the 386 environment. Xenix-386 is out, albeit with a 286 kernel, and non-Unix types may be willing to wait for an MS-DOS. But The Software Link of Atlanta is promising to deliver a native-mode multitasking operating system for the 386 that will be compatible with and replace DOS. And DRI keeps polishing its Concurrent DOS for PC machines—the version now in beta is getting good marks. Well, we'll see.

On the flight to Redmond I worked a crossword puzzle. One clue was "one who takes drugs." The answer: addict. Yes, it's heartening to see even crossword puzzle writers getting in on the act and analyzing the drug problem as thoroughly as Congress and the television networks have.

My cousin Corbett thinks *DDJ* ought to join the campaign. Addiction, he says, is a serious problem among programmers, and it's spreading rapidly. Corbett has shown me frightening statistics on the number of programmers who only get up from their keyboards during the compile phase and urges me to consider what will happen to these code junkies when they get their hands on 386 machines. Laboratory pigeons on a killer reinforcement schedule. Gruesome.

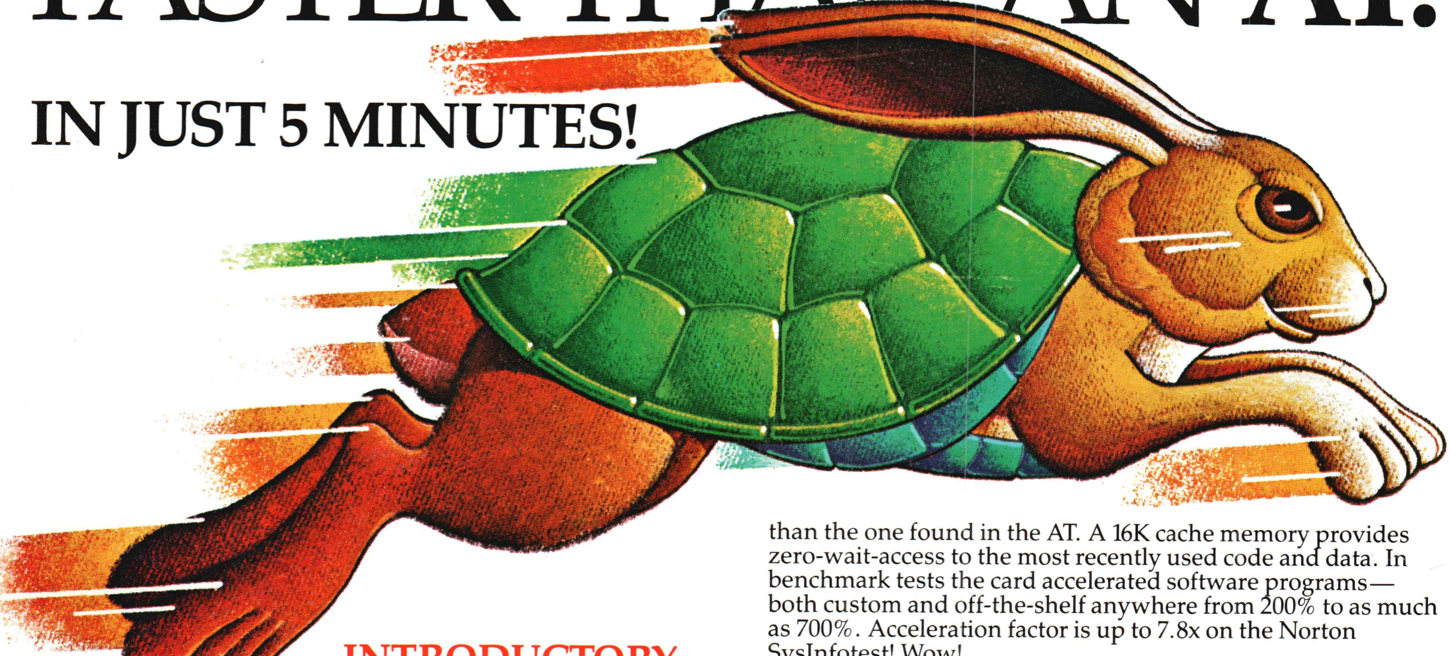
I'd been wondering for months what was behind Fast Willie's editorial in *PC Tech Journal* begging Intel to hold off on releasing the 386. Now I understand: Willie was doing his bit to fight programming addiction.

Corbett, who abhors mice and windows, spent some time recently programming in Smalltalk and concluded that it's nonaddictive. He is currently negotiating with a software vendor to develop a rehabilitating object-oriented programming environment, the first version of which will be called MethodOne.

Michael Swaine

Michael Swaine
editor-in-chief

MAKE YOUR IBM PC FASTER THAN AN AT.



IN JUST 5 MINUTES!

INTRODUCTORY OFFER $395

DON'T TAKE OUR WORD FOR IT. USE IT FOR 60 DAYS. IF YOU ARE NOT TOTALLY SATISFIED RETURN IT FOR A FULL REFUND.

It sounds great; the idea of a speedup board that you can just plug right in as easily as putting bread in a toaster. How wonderful to be able to convert a PC or XT to a $4000 AT without the expense. But even when you get ready to spend $395.00 you want to be sure your choice is the very best.

Here at PCSG we sell our IBM PC disk access speedup software by the thousands. But software doesn't do anything about speeding up the microprocessor (or CPU) speed. As you know the microprocessor is the brain of the computer that controls all the operations like screen updates and calculations like a spreadsheet makes.

Faster and smarter than an AT – PCSG guarantees it.

We wanted to offer a speedup card that would be the complement to our disk speedup software, (incidentally included at no extra charge.) We wanted it to be literally the most advanced, compatible and feature rich board available today. We could only be satisfied with a board that was the finest example of the engineering art.

There is no question we have met our every objective by developing and manufacturing the *BREAKTHRU 286* card. This is the best designed and most functional speed up card available today. We guarantee it.

HERE IS WHAT MAKES IT SO SPECIAL.

First, it installs so easily. It is a half slot card, only five inches in length. You don't even have to give up a full slot. What's more, unlike competing products it works in the Compaq and most clones. The instructions are so simple we considered showing a picture of a child putting it in. Easy diagrams show how you just place the card in an open slot, remove the original processor and connect a single cable. There is no software required. From that moment you are running faster than an AT.

Second, it is advanced. The *BREAKTHRU 286* replaces the CPU of the PC or XT with an 80286 microprocessor that is faster than the one found in the AT. A 16K cache memory provides zero-wait-access to the most recently used code and data. In benchmark tests the card accelerated software programs—both custom and off-the-shelf anywhere from 200% to as much as 700%. Acceleration factor is up to 7.8x on the Norton SysInfotest! Wow!

Third, you have full compatibility. All existing system RAM, hardware, and peripheral cards can be used without software modification. It operates with LAN and mainframe communication products and conforms to the Lotus/Intel/Microsoft Expanded Memory Specification (EMS). Software compatibility is virtually universal.

Fourth, it is the best there is. There are several other boards on the market. Some are priced about the same as the *BREAKTHRU 286* and some are cheaper. We at PCSG have compared them all, but there simply was no comparison. What we discovered is that many cards being sold offer only a marginal speed up in spite of their claims. We found some to be merely versions of the obsolete 8088 or 8086, and others to be just poorly engineered. The 8MHz *BREAKTHRU 286* is unequivocally the best executed and most completely reliable speedup board manufactured today.

PCSG has since early 1983 dominated the lap portable market with ROM software such a Lucid spreadsheet and Write ROM that reviewers rated as excellent. We were proud to successfully enter the IBM PC market last year with disk access speedup software. Now we are so pleased with the *BREAKTHRU* speedup card. We use them on our own PC's to make them faster than AT's. We are really excited about this product.

PCSG makes the unabashed statement that the *BREAKTHRU 286* card represents more advanced technology than boards by Orchid, Quadram, Victor, Mountain, P.C. Technologies, Phoenix . . . we could go on.

But an ad can't let you experience it for yourself. That's why we sell the *BREAKTHRU 286* on a 60 day trial. If you aren't completely satisfied return it within 60 days for a full refund. It is priced at $395. Call today with your MasterCard, Visa, American Express or COD instructions and we will ship your card the very next day.

Circle **no. 224** on reader service card.

PCS

PERSONAL COMPUTER SUPPORT GROUP

11035 Harry Hines Blvd. #207 • Dallas, Texas 75229

214-351-0564

master charge · American Express Cards Welcome · VISA

Why more than half a million people are using Turbo Pascal

Because Turbo Pascal is faster than any other Pascal compiler—it's become the industry standard. And because Turbo Pascal is backed by a complete range of "tool boxes" that give you most of the programming tools you'll ever need.

- Turbo Pascal® 3.0 with 8087 and BCD
- Turbo Database Toolbox™
- Turbo Graphix Toolbox®
- Turbo Tutor® 2.0
- Turbo Editor Toolbox™
- Turbo GameWorks®

The Turbo Pascal family is never static, but is continuously expanding, with new products like Turbo Editor Toolbox and Turbo Gameworks.

> "Language deal of the century . . . Turbo Pascal
> *Jeff Duntemann, PC Magazine*
>
> Turbo Pascal has got to be the best value in languages on the market today
> *Jerry Pournelle, BYTE Magazine*
>
> This compiler, produced by Borland International, is one of the best programming tools presently available for the PC
> *Michael Covington, PC Tech Journal*"

The secret of software success is not merely low price, but top quality allied with complete documentation, like our 400-page reference manual.

All of which are some of the reasons why Turbo Pascal is clearly the leader, and the recipient of awards like *PC Week's* "Product of the Year" and *PC Magazine's* "Award for Technical Excellence."

A whole family of tools

Success breeds success, so the Turbo Pascal family has flourished. Your choices now include:

☐ **Turbo Pascal 3.0** combines the fastest Pascal compiler with an integrated development environment, *and* 8087 math co-processor support *and* Binary Coded Decimals.

☐ **Turbo Database Toolbox** a perfect complement to Turbo Pascal. It includes a complete library of Pascal procedures that allows you to search and sort data and build powerful database applications.

☐ **Turbo Graphix Toolbox** includes a library of graphics routines for Turbo Pascal programs. Lets even beginning programmers create high-resolution graphics with an IBM,® Hercules,™ or compatible graphics adapter. Does complex business graphics, easy windowing, and stores screen images to memory.

☐ **Turbo Tutor 2.0** teaches you step-by-step how to use Turbo Pascal, with commented source code for all program examples on diskette.

An offer you can't refuse

You can save $119.75 when you choose the Turbo Jumbo Pack. Six different Turbo Pascal products for only $299.95.

You get Turbo Pascal 3.0 *and* Turbo Editor Toolbox *and* Turbo Tutor 2.0 *and* Turbo Graphix Toolbox *and* Turbo GameWorks *and* Turbo Database Toolbox!

So act now—rush to your dealer or call us today!

NEW! ***Amazing value! Turbo Editor Toolbox includes MicroStar, a full-blown editor that also does windows!*** You get ready-to-compile source code and a 200-page manual that tells you how to integrate the editor procedures and functions into your programs. Also includes MicroStar, a complete editor with full windowing capabilities. (You could pay $100.00 or more for a program like MicroStar, but you get it free as part of our Turbo Editor Toolbox.) You can also use Turbo Editor (which of course integrates with Turbo Lightning®) to build your own word processor!

NEW! ***Turbo GameWorks gives you the games you can play, write, rewrite, bend and amend!*** Turbo GameWorks reveals the secrets and the strategies of game design. You're given source code, a 200-page manual, and the insight needed to write and customize your own irresistible games.

Also includes ready-to-play Chess, Bridge, and Go-Moku—an ancient Japanese game that can divert you from reality for hours on end.

NEW! ***TNT* A DYNAMITE DEAL!***
**Turbo 'N Tutor*
Turbo Pascal *and* Turbo Tutor for only $125.00. You save nearly $15.00, and have everything you need to get up and go with Turbo Pascal!

YES! I want the best!

To order by phone, or for a dealer nearest you,
call (800) 255-8008
in CA call (800) 742-1133

Copies	Product	Price	Totals
___	Turbo Pascal 3.0 w/8087 & BCD**	$ 99.95	$ ______
___	Turbo Pascal for CP/M-80	69.95	$ ______
___	Turbo Database Toolbox	69.95	$ ______
___	Turbo Graphix Toolbox*	69.95	$ ______
___	Turbo Tutor 2.0	39.95	$ ______
___	Turbo Editor Toolbox*	69.95	$ ______
___	Turbo GameWorks*	69.95	$ ______
___	Turbo Pascal & Turbo Tutor	125.00	$ ______
___	Turbo Jumbo Pack*	299.95	$ ______
	Outside USA add $10 per copy CA and MA res. add sales tax		$ ______
	Amount enclosed		$ ______

Prices include shipping to all US cities.

Carefully describe your computer system:

Mine is: __ 8-bit __ 16-bit

I use: __ PC-DOS __ MS-DOS __ CP/M-80 __ CP/M-86

My computer's name and model is: ______________________

The disk size I use is: ☐ 3½" ☐ 5¼" ☐ 8"

Payment: VISA MC Money Order Check

Credit card expiration date ___/___

Card # | | | | | | | | | |

TP11

Name: ______________________

Shipping Address: ______________________

City: ______________________

State: ______________ Zip: ______________

Telephone: ______________________

CODs and purchase orders WILL NOT be accepted by Borland. Outside USA make payment by bank draft, payable in US dollars drawn on a US bank.

NOT COPY PROTECTED
60-DAY MONEY-BACK GUARANTEE

If within 60 days of purchase this product does not perform in accordance with our claims, call our customer service department and we will gladly arrange a refund.

All prices are suggested list prices and are subject to change without notice.

Minimum System Requirements:
Turbo GameWorks, Turbo Graphix Toolbox, Turbo Tutor, & Turbo Editor Toolbox—192K. All other products, 128K.

*IBM PC, PCjr, AT, XT, and true compatibles.

**16-bit only.

BORLAND
INTERNATIONAL
4585 SCOTTS VALLEY DRIVE
SCOTTS VALLEY, CA 95066
(408) 438-8400 TELEX: 172373

Vive la différence

Circle **no. 161** on reader service card.